
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**EFFICIENT CALCULATION OF EARTH PENETRATING PROJECTILE TRAJECTORIES**

by

Daniel F. Youch

September 2006

Thesis Advisor: Joshua Gordis

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** September 2006 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** Efficient Calculation of Earth Penetrating Projectile Trajectories | | **5. FUNDING NUMBERS** | |
| **6. AUTHOR(S)** Daniel F. Youch | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** | |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** | |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | **12b. DISTRIBUTION CODE** | |


**13. ABSTRACT (maximum 200 words)**

Currently, two methods exist to determine trajectory of a ballistic penetrator: Poncelet Analysis and Differential Area Force Law (DAFL) methods. An exact solution for the Poncelet Equation exists; making for easy computation. However, the one dimensional nature of the equation fails to capture the intricate three-dimensional nature of real world ballistic penetrator trajectories. The DAFL methods employ empirically derived stress algorithms to calculate the forces acting on a differential area of a projectile. These stresses are then used to determine the forces and moments acting on the differential areas. These forces and moments are then used to solve the equations of motion to determine the trajectory of the ballistic penetrator. The DAFL methods accurately capture the three dimensional nature of the penetrator's trajectory, but are computationally intensive which make them slow.

The Integrated Force Law (IFL) method combines the computational ease of the Poncelet Analysis with the accuracy of the DAFL methods. In IFL, the projectile shape is modeled as a polynomial. The stress algorithms used in the DAFL methods are then numerically integrated over the top and bottom surfaces of the projectile to determine the force and moment acting on the top and bottom half of the weapon. These two forces and moments are then used to solve the equations of motion. J-hook trajectories are solved in less than 40 seconds and stable trajectories are solved in less than three seconds.


| **14. SUBJECT TERMS** Ballistic, Penetration, Simulation | | | **15. NUMBER OF PAGES** 119 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UL |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

# EFFICIENT CALCULATION OF EARTH PENETRATING PROJECTILE TRAJECTORIES

Daniel F. Youch
Lieutenant Commander, United States Navy
B.S., Temple University, 1995

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MECHANICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2006**

Author: Daniel F. Youch

Approved by: Joshua Gordis
Thesis Advisor

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

Currently, two methods exist to determine trajectory of a ballistic penetrator: Poncelet Analysis and Differential Area Force Law (DAFL) methods. An exact solution for the Poncelet Equation exists; making for easy computation. However, the one dimensional nature of the equation fails to capture the intricate three-dimensional nature of real world ballistic penetrator trajectories. The DAFL methods employ empirically derived stress algorithms to calculate the forces acting on a differential area of a projectile. These stresses are then used to determine the forces and moments acting on the differential areas. These forces and moments are then used to solve the equations of motion to determine the trajectory of the ballistic penetrator. The DAFL methods accurately capture the three dimensional nature of the penetrator's trajectory, but are computationally intensive which make them slow.

The Integrated Force Law (IFL) method combines the computational ease of the Poncelet Analysis with the accuracy of the DAFL methods. In IFL, the projectile shape is modeled as a polynomial. The stress algorithms used in the DAFL methods are then numerically integrated over the top and bottom surfaces of the projectile to determine the force and moment acting on the top and bottom half of the weapon. These two forces and moments are then used to solve the equations of motion. J-hook trajectories are solved in less than 40 seconds and stable trajectories are solved in less than three seconds.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

I would like to acknowledge my wife, Alisha, for all of her effort toward keeping life normal while I was working on my thesis. Additionally, I would like to acknowledge my advisor, Joshua Gordis, for his keen and timely guidance.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. BACKGROUND

Accurate prediction of an earth penetrating projectile's trajectory while traveling through a media is vital to effective ordinance employment. If the trajectory of these projectiles can be accurately predicted, the number of projectiles employed against a target can be reduced since it is precisely known where each projectile will finish its trajectory. However, there are many factors that complicate predicting these trajectories.

Projectiles traveling through the ground do not always travel in a straight path. Under certain conditions, the trajectory will take on a curvilinear shape, much like "J". This J-Hook phenomenon makes it difficult to determine the trajectory of the weapon. Additionally, the entire weapon does not stay in contact with the surrounding soil. This condition is known as Wake Separation. A stress is exerted on the projectile only where the soil is in contact with the surface of the projectile. Wake separation is a complicated; little understood phenomena that greatly impacts the trajectory a projectile will take. Finally, there is Trajectory Direction Reversal. At some critical incidence angle a projectile that would normally execute a J-Hook trajectory no longer travels back toward the surface, but dives away from the surface, driving the projectile much deeper than is expected. All of these issues make accurate prediction of a penetrator's trajectory very difficult.

The most limiting factor toward accurate prediction of a projectile's trajectory is computing time. Methods currently in use that predict reasonably accurate projectile trajectories employ large models of the projectile. These large models require a great deal of computing time to solve for the trajectory. This thesis investigates if it is possible to replace the large projectile models currently in use with a smaller, less complicated model in the hopes that this smaller model will reduce computing time without sacrificing trajectory accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. PREVIOUS RESEARCH

## A. PONCELET ANALYSIS

Investigation into projectile impact began with Poncelet in the latter half of the 19th century. Poncelet was able to empirically determine that the equation that describes the acceleration a projectile experiences while traveling through a media is given by Eqn. (1).[1]

$$-\frac{dV}{dt} = A + BV^2 \tag{1}$$

where the penetration constants, *A* and *B*, are functions of both the projectile mass and the media through which the projectile is traveling. Eqn. (1) can be integrated to find the penetration depth as a function of time Eqn. (2).[2]

$$y(t) = y_o + \frac{\ln\left\{\cos(\sqrt{AB}(t-t_o)) + V_o\sqrt{B/A}\sin(\sqrt{AB}(t-t_o))\right\}}{B} \tag{2}$$

In order to solve for the depth of the projectile, constants *A* and *B* must be determined. In the 1970's Sierakowski et al.[3] at the University of Florida found these constants using empirical data. The constants were solved for near normal impact of small projectiles. Once they had solved for the penetration constants in all three dimensions, they developed a 3 dimensional model based on the empirically determined constants. Their 3 dimensional model was then compared to empirical normal impact data.

This research yielded good agreement between the theoretical model and experimental data for normal impact. However, there are some significant drawbacks to this method. First, the Poncelet constants are different for each projectile shape, projectile mass, and soil type. Therefore, the penetration constants must be empirically determined for each different projectile and soil type. Secondly, in order to solve for the position, 12 coupled differential equations must be solved (3 force equations, 3 moment equations,

[1] R.L. Sierakowski, L.E. Malvern, J.A. Collins, J.E. Milton, and C.A. Ross, (1977). "Penetrator Impact Studies of Soil/Concrete," University of Florida, Gainesville, Florida. p. 17.

[2] Ibid.

[3] Ibid.

and 6 velocity equations). While good results were obtained, this method is not conducive to rapid trajectory calculation of projectiles of various shapes.

## B. DIFFERENTIAL AREA FORCE LAW (DAFL) METHOD

A few years later, Bernard and Creighton[4],[5] applied the principles of soil mechanics to determine the forces acting on the surface of the projectile as it moves through soil. The projectile was discritized into many small differential areas. The forces and moments acting on these areas are summed to determine the total force and moment acting on the projectile. This method of determining the overall forces and moments acting on the projectile by summing the forces and moments acting on discrete areas is known as the Differential Area Force Law (DAFL) method.

In order to apply these formulations, large Differential Area Force Law (DAFL) models of the projectile were needed. The DAFL model is constructed by replacing the smooth curve of the projectile with a series of differential areas. This converts the projectile from a smooth surface to a multi-faceted surface. The stress at the center of these differential areas was calculated using the available stress formulations then multiplied by the differential area. The resulting force vector for each differential area was then summed to determine the force acting on the projectile Eqn. (3).

$$\vec{F} = \sum_{i=1}^{m} \vec{n}_i \sigma_i dA_i \tag{3}$$

The moment acting on the projectile is calculated in a similar fashion is given by Eqn. (4).

$$\vec{M} = \sum_{i=1}^{m} \vec{r}_i \times \vec{F}_i \tag{4}$$

The DAFL method has the advantages of being able to accurately describe the 3 dimensional motions of the projectile while solving only two vector differential Eqns. (5) and (6) instead of the twelve differential equations employed by Sierakowski.

---

[4] R.S. Bernard, and D.C. Creighton, (1978). "Non-Normal Impact and Penetration: Analysis for Hard Targets and Small Angles of Attack," Technical Report SL-78-14, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 9-13.

[5] R.S. Bernard, and D.C. Creighton, (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 9-12.

$$\ddot{\vec{a}} = \frac{\vec{F}}{m} \tag{5}$$

$$\ddot{\vec{\theta}} = \frac{\vec{M}}{I} \tag{6}$$

In this method, the forces and moments acting on the weapon are not dependent on projectile and soil specific constants. The stress experienced by a differential area on the projectile is a function of projectile speed and soil penetrability. Soil penetrability is quantified by the soil number, SNUM, with less penetrable soils having a smaller SNUM. A sample of soil types and their corresponding SNUM are given in Table 1.

| Soil Type | SNUM |
|---|---|
| Hard Sand | 4 |
| Sand | 8 |
| Elgin Sand | 11 |
| Soft Clay | 35 |

Table 1. Soil Type and Corresponding SNUM[6]

Different constants need not be calculated for each projectile, making it more adaptable than the earlier Poncelet attempts. However, a large finite element model is still required leading to long computational times.

A third model that combines elements of both the Poncelet analysis and the DAFL method was developed by Young (1972). In this model, Young uses his empirically derived equation for predicting penetration depth Eqn. (7).[7]

$$D = 0.00178S(0.2W^{.4})N(W/A)^{.7}(V-100) \tag{7}$$

[6] D.C. Creighton, (1988). "Ricochet and Stability of MK82, MK83 and MK84 General Purpose Bombs in Soil," Miscellaneous Paper SL-88-1, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 20.

[7] C.W. Young, (1997). "Penetration Equations," Contractor Report SAND97-2426, Applied Research Associates, Inc., Albuquerque, New Mexico. pp. 5-6.

where $S$ is the SNUM, $W$ is the projectile weight, $A$ is average cross sectional area, $N$ is local nose performance coefficient, and $V$ is the initial projectile velocity.

This depth is then used to calculate the average axial force acting on the projectile Eqn. (8).[8]

$$F_a = m\frac{V^2}{2D} \tag{8}$$

Lateral force, $F_{\alpha_i}$, acting on a differential area is given by Eqn. (9).[9]

$$F_{\alpha_i} = \frac{\varepsilon\varepsilon' A_{S_i} F_a \alpha_i}{0.06d^2} \tag{9}$$

where $\varepsilon$ is the cratering factor at weapon entrance, $\varepsilon'$ is the cratering factor at weapon exit, $A_{S_i}$ is the surface area of the differential area, $\alpha_i$ is the local angle of attack, and $d$ the projectile diameter.

Note that the lateral force is still a function of the average acceleration. These forces are then used to solve the same equations of motion Eqns. (5) and (6) that are used by the DAFL method.

[8] C.W. Young, (1998). "Simplified Analytical Model of Penetration with Lateral Loading," Contractor Report SAND98-2426, Applied Research Associates, Inc., Albuquerque, New Mexico. p. 6.

[9] Ibid., p. 8.

# III. INTEGRATED FORCE LAW METHOD (IFL)

## A. COORDINATE SYSTEM

Two coordinate systems were used in this investigation. These coordinate systems are pictured in Figure 1 and Figure 2. The inertial coordinate system is depicted in Figure 1 and the weapon based coordinate system shown in Figure 2.

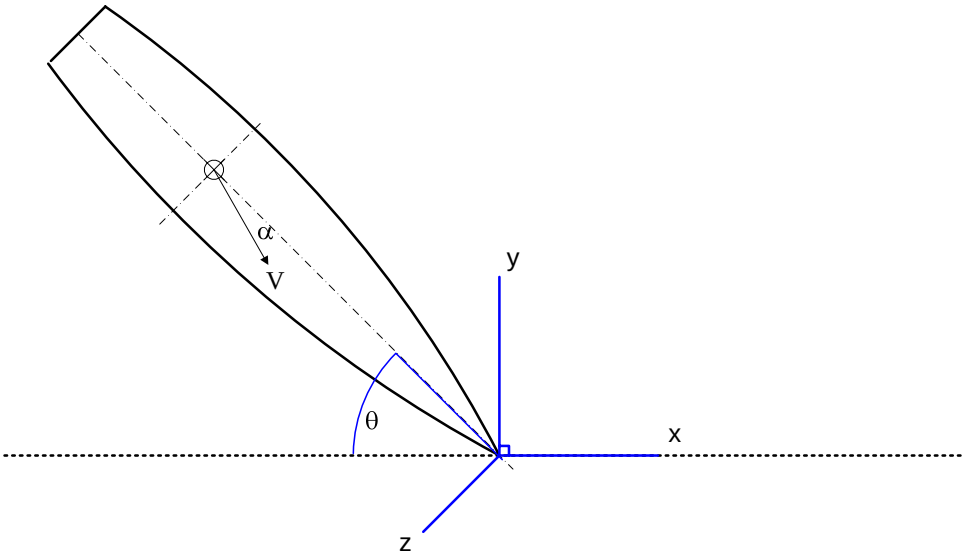


Figure 1. Inertial Coordinated System

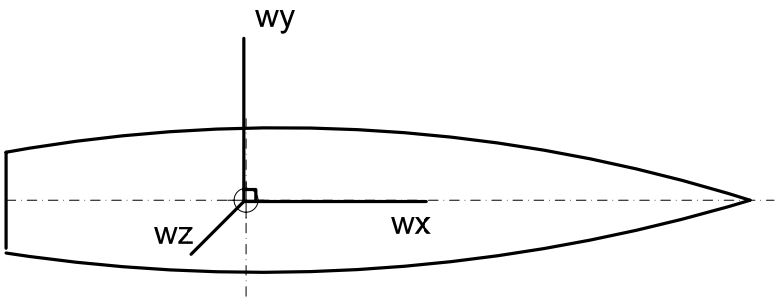


Figure 2. Weapon Coordinate System

The origin of the inertial system is fixed to the point where the tip of the projectile impacts the soil. The x-axis is along the soil interface. Positive incidence angle, $\theta$, is measured counter clockwise from the weapon centerline to a line parallel to the x-axis that passes through the projectile CG. The angle of attack, $\alpha$, is the angle between the CG velocity vector and the projectile centerline. Positive $\alpha$ is measured clockwise from

projectile centerline to the CG velocity vector. If the projectile centerline makes an angle $\theta$, with a line parallel to the inertial x-axis, the matrix to transform the CG velocity and the forces and moments acting on the projectile from inertial to weapon coordinates is given by Eqn. (10).

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{10}$$

where $\theta$ is the incidence angle.

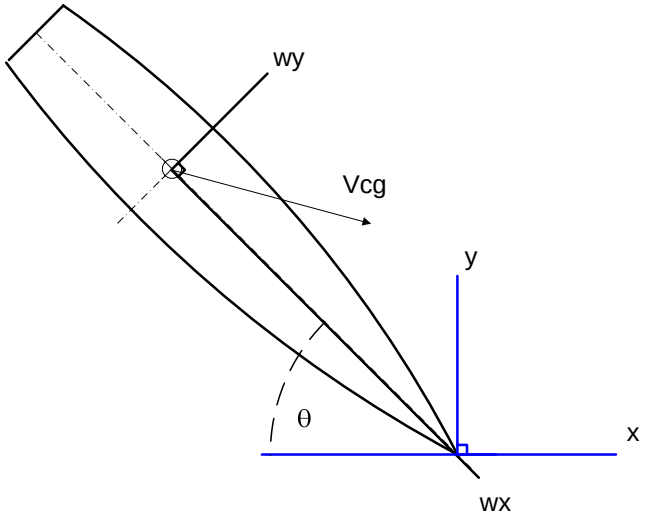


Figure 3. Inertial to Weapon Frame Transformation Geometry

## B. PROJECTILE MODEL

The MK-84 bomb was the projectile modeled. The MK-84 nosecone is shaped like a tangent ogive The MK-84 physical characteristics and model characteristics are listed in Table 2. Note that the Moment of Inertia for the ISAAC II model is less than the actual moment of inertia of the projectile. This reduction in the moment of inertia is necessary for the ISAAC II stress formulation to reproduce the J-Hook Phenomena that the DAFL method produces. This reduction in the moment of inertia for the ISAAC II formulation is most likely needed to compensate for an inaccurate projectile model and an overly simplistic wake separation model. These issues will be addressed in the RESULTS and ANALYSIS AND DISCUSSIONS sections. It is worth noting here, however, that if the moment of inertia used for the PENCURV reduced by a factor of twenty, there is no corresponding increase in the amount of weapon rotation.

| Projectile Physical Data | | | |
|---|---|---|---|
| | Mk-84 Projectile | PENCURV Model | ISAAC II Model |
| Overall Length (in) | 101 | 101 | 101 |
| Length Nose to CG (in) | 59.5 | 59.5 | 59.5 |
| Length CG to Tail (in) | 41.5 | 41.5 | 41.5 |
| Maximum Radius (in) | 9 | 9 | 9 |
| Weight (lbf) | 1890 | 1890 | 1890 |
| Moment of Inertia $\left(\text{lbf}/\text{in}^4\right)$ | $1.367\text{x}10^6$ | $1.367\text{x}10^6$ | $6.835\text{x}10^4$ |

Table 2. Physical and Model Characteristics for MK-84 Projectile[10]

In the IFL method, the shape of the projectile is approximated by a continuous polynomial. Once this polynomial has been determined, the surface area of the projectile can be approximated by rotating the polynomial about the y-axis. The polynomial will be determined by performing a least squares fit to the coordinates that describe the projectile curve. The nosecone of the MK-84 bomb is described as a tangent ogive. A tangent ogive is a portion of a circle of radius $\rho$ given by Eqn. (11) shown in Figure 4.

[10] D.C. Creighton, (1988). "Ricochet and Stability of MK82, MK83 and MK84 General Purpose Bombs in Soil," Miscellaneous Paper SL-88-1, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 17.

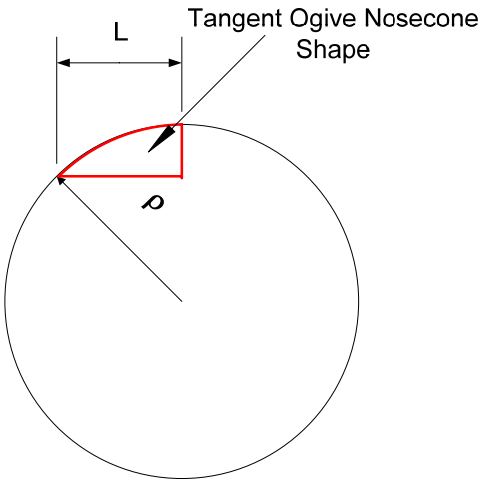


Figure 4. Tangent Ogive Geometry

To facilitate transformation between the inertial frame and the weapon frame, the projectile CG is placed at the origin of the weapon frame. The maximum radius of 9 inches occurs at the CG. The shape of the projectile forward of the CG is modeled as a tangent ogive with a length of 59.5 inches and a maximum radius of 9 inches. Eqn. 11 is used to determine the ogive radius as a function of the length and radius of the nosecone. Eqn. 12 is used to calculate the y coordinates of the projectile. The x input to Eqn. (12) will be the roots of the Chebyshev polynomial. The roots of the Chebyshev polynomials are routinely used as the x coordinates for interpolating polynomials because they can reduce the degree of the polynomial without a significant increase in the error associated with the approximation.[11] Eqns. (11)[12] and (12)[13] are used to calculate the y coordinates of the projectile shape.

[11] R.L. Burden, J.D. Faires, (2005). "Numerical Analysis, 8th Edition," Thomson Brooks/Cole, Belmont, California. p. 510.

[12] G.A. Crowell, Sr. (1996). "The Descriptive Geometry of Nose Cones," Miscellaneous Paper.

[13] Ibid.

$$\rho = \frac{R^2 + L^2}{2R} \tag{11}$$

$$y = \sqrt{\rho^2 - (x - L)^2} + (R - \rho) \tag{12}$$

where $\rho$ is the ogive radius, $R$ is the projectile radius at the base of the nosecone and $L$ is the length of the nosecone.

The roots of the Chebyshev polynomial are given by Eqn. (13).[14] Eqn. (13) can be scaled to any interval by Eqn.(14).[15]

$$x_{n-1} = \cos\left(\frac{(N+1+1/2-n)\pi}{N+1}\right) \tag{13}$$

$$x_{n-1} = \frac{1}{2}\cos\left((b-a)\frac{(N+1+1/2-n)\pi}{N+1} + a + b\right) \tag{14}$$

where N is the total number of roots desired, n is the current root desired, a is the lower limit of the interval and b is the upper limit of the interval. The range of n is from 1 to N+1.

The afterbody coordinates are determined in a similar fashion, but are scaled from zero to 106.5 inches. Once the coordinates of the entire projectile have been found, a least squares fit of the points is used to find the degree 3 polynomial, *f(x)* that describes the shape of the top half of the projectile. The bottom half of the projectile, *g(x)*, is given by *g(x)=-f(x).* Since the shape of the weapon is described by a polynomial, the functions *f(x)* and *g(x)* are continuous and continuously differentiable of the range of interest.

## C. PHYSICS OF THE J-HOOK

Consider the two projectiles, one with a straight afterbody and one with a flared afterbody, shown in Figure 5 below.

[14] R.L. Burden, J.D. Faires, (2005). "Numerical Analysis, 8th Edition," Thomson Brooks/Cole, Belmont, California. p. 505.

[15] Ibid., p. 508.

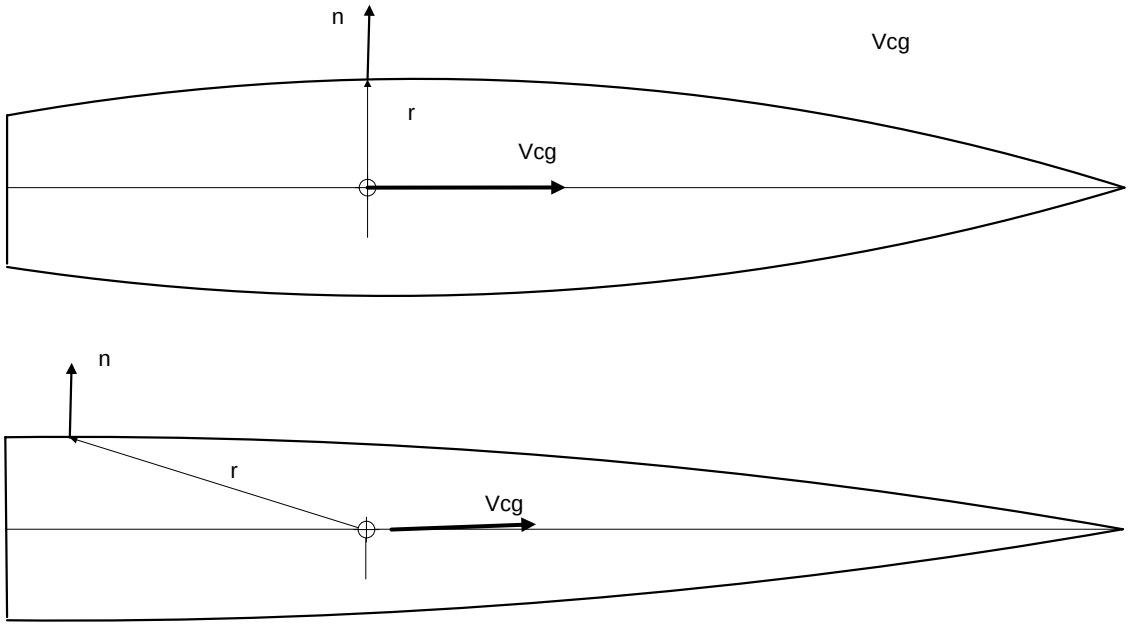


Figure 5. Tapered and Straight Afterbody Projectiles

Both are about to impact the soil with zero angle of attack. For this discussion, the effect of the interface will be ignored. This effect will be discussed later. At impact, both projectiles will be imparted with a counter clockwise rotation and the angle of attack will become slightly positive.

Recall that a force on the surface is only imparted when the dot product of the velocity and the outward surface normal is greater than zero. Consider only the straight afterbody case. As the angle of attack becomes slightly positive, only a small portion of the upper surface has a dot product of less than zero with respect to the CG velocity. As a result, there is only a slight imbalance in the resultant moment on the projectile. As a result, the projectile rotates only slightly as it moves through the soil.

Now consider the tapered after body. As the projectile moves through the soil, the angle of attack becomes slightly negative (above the centerline of the weapon) as shown in Figure 6.

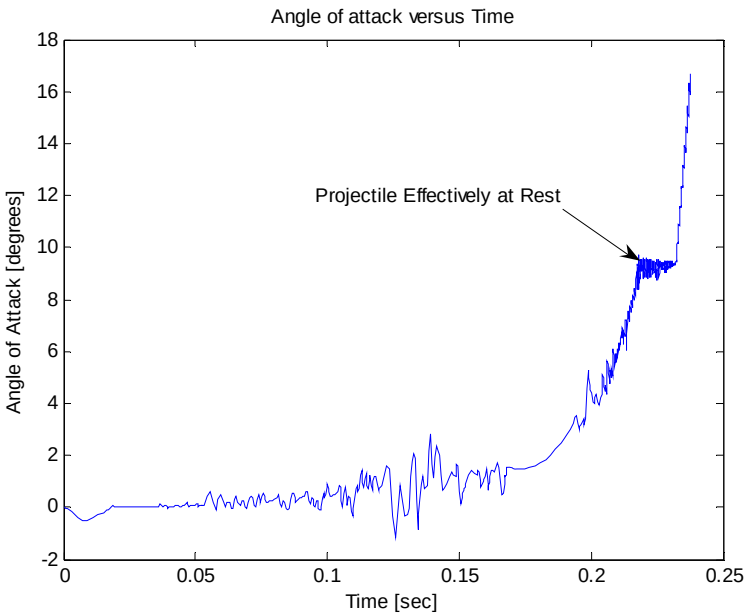


Figure 6. Angle of Attack versus Time for MK-84

As the angle of attack becomes more negative, less surface area on the bottom of the projectile is in contact with soil. On the top surface of the projectile, more surface area becomes in contact with the soil. As a result, the force distribution shown in Figures 7 and 8 results. Figures 7 and 8 are for an angle of attack of -2 degrees. The force acting on the top half of the combines with the force on the lower half of the weapon to produce a force with a negative axial component and a negative transverse component. As a result, the force is acting in the opposite direction of velocity and the weapon continues to slow.

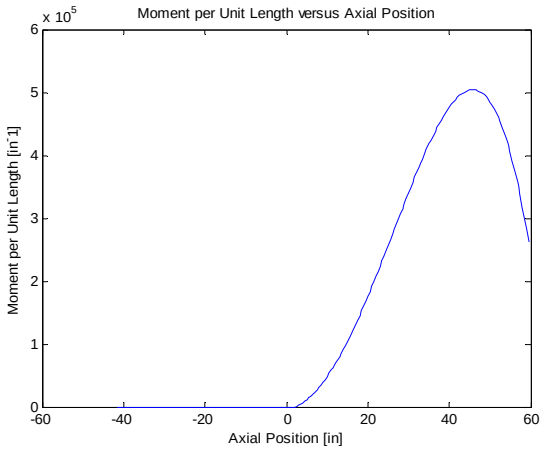


Figure 7. Forces on Top Half of Weapon from 2 Degree Angle of Attack

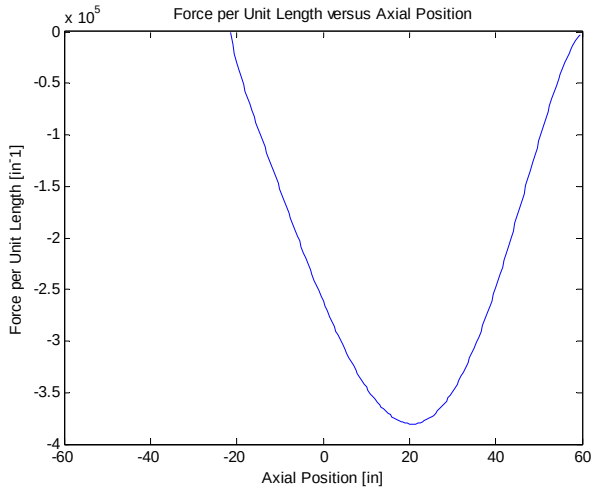


Figure 8. Forces on Bottom Half of Weapon from 2 Degree Angle of Attack

The force distribution seen in Figures 7 and 8 produce the moment distributions seen in Figures 9 and 10.

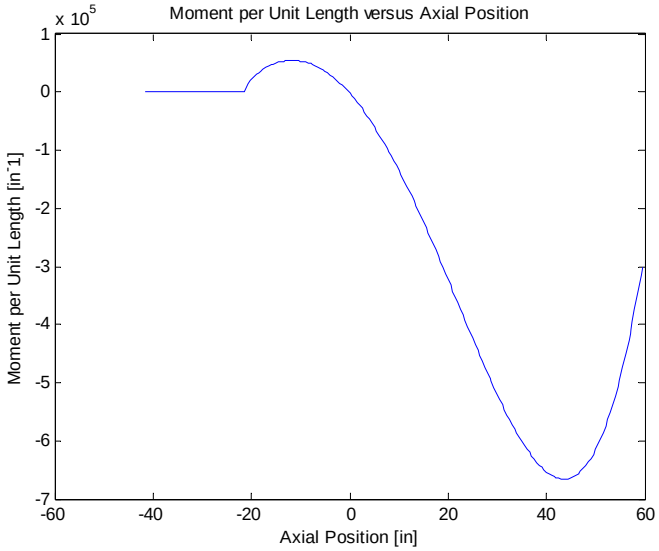


Figure 9. Moment on Top Half of Weapon from a 2 Degree Angle of Attack

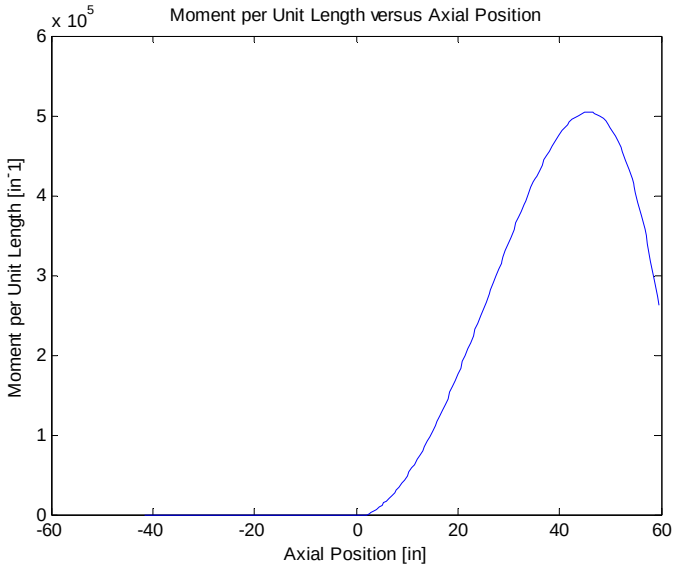


Figure 10. Forces on Bottom Half of Weapon from 2 Degree Angle of Attack

The forces acting on the bottom half of the weapon produce only a positive moment. The forces on the top half of the weapon, however, produce a negative moment forward of the CG and a positive moment aft of the CG. The magnitude of the negative moment from the top half of the weapon is only slightly larger than the magnitude of the positive moment from the bottom half of the weapon. The resultant negative moment, however, is smaller than the positive moment produced by the forces acting aft of the CG on the top half of the weapon. The overall resultant moment acting on the weapon is positive causing the weapon to rotate in a counter clockwise direction. This small positive moment is what accounts for the J-Hook phenomena seen in projectiles with tapered after bodies.

## D. FORCE AND MOMENT CALCULATIONS USING THE IFL METHOD

This thesis investigates whether or not it is possible to solve the trajectory of a projectile quickly and accurately without resorting to a DAFL model. Two different stress formulations were explored. First was the PENCURVE stress formulation Eqn. (15)[16] that is used when a stable, non J-Hook trajectory is expected. The PENCURVE stress formulation should be employed in all cases except where a J-Hook is expected and

[16] M.D. Adley, B.P. Berger, and D.C. Creighton, (1994). "Two-Dimensional Projectile Penetration Into Curvilinear Geologic/Structured Targets: User's Guide for PENCURV-PC," V1.5, Instruction Report SL-94-1 US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 10.

then only if the entire trajectory is contained in a homogenous soil half space. A J-Hook trajectory is only expected when the projectile has a tapered afterbody and the soil half space is not stable. Topsoil, sand and clay are considered unstable half spaces where subsoil, rock, and concrete are considered stable half spaces. The ISAAC II stress formulation Eqns. (16)-(19)[17] was also explored. ISAAC II is used only when the J-hook trajectory path is expected. ISAAC II should only be used in a homogenous soil half space. Eqn. (15) yields the magnitude of the normal force acting at a point on the surface of a projectile when using the PENCURV stress formulation

$$\sigma = \frac{0.625\mu}{Sr_p}\sqrt{\frac{v_n}{v}} + \frac{5.19v}{S}\sqrt{\beta r_p \frac{v_n}{v}} + \frac{5.9\gamma Z}{S^2}\frac{v_n}{v} \tag{15}$$

where $\mu, \beta, \gamma$ are experimentally determined curve fit parameters, $S$ is SNUM, $v_n$ is the component normal to the projectile surface of $v$, $v$ is the local velocity at a point on the projectile, $Z$ is the depth of the point on the projectile and $r_p$ is the local projectile radius.

$$\sigma_a = \frac{2}{\pi}\left( \frac{\mu}{S \cdot N \cdot D} + \frac{2\beta Z}{S^2 \cdot N^2} + \frac{V_{localx}\sqrt{\beta m\, {}^3\!/_7}}{S \cdot D} \right) \tag{16}$$

where $V_{localx}$ is the velocity of a point in the **wx** direction in the projectile frame, $m$ is the projectile mass and $D$ is the local projectile diameter.

$$\sigma_t = \frac{4}{\pi}\left( \frac{\mu}{SNUM \cdot D} + \frac{2\beta Z}{SNUM^2} + \frac{V_{localy}\sqrt{\beta m\, {}^3\!/_7}}{SNUM \cdot D} \right) \tag{17}$$

$$N = \frac{0.508}{\sqrt{\sin\theta}} \tag{18}$$

where $N$ is the local nose performance coefficient and $\theta$ is the angle formed by a line tangent to the nosecone at a point and the projectile centerline.

$$\sigma_\perp = \sqrt{\sigma_a^2 + \sigma_t^2} \tag{19}$$

[17] I M.D. Adley, B.P. Berger, and D.C. Creighton, (1994). "Two-Dimensional Projectile Penetration Into Curvilinear Geologic/Structured Targets: User's Guide for PENCURV-PC," V1.5, Instruction Report SL-94-1 US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 10. and Stress Equation, p. 12.

Eqn. (19) yields the magnitude of the normal force acting at a point on the surface of a projectile when using the ISAAC II stress formulation.

Both formulations assume that the stress that a projectile experiences from traveling through a media is always normal to the surface of the projectile. Since the resulting force acts normal to the surface, and the stress formulations employed yield the magnitude the product of stress and area is multiplied by the inward normal vector.

Another assumption is that the stress generated is a function of the velocity of the point on the projectile where the stress is experienced. Since stress is a function of velocity and it occurs normal to the surface of the projectile, stress only occurs if the angle between the surface normal and the velocity vector is less than 90 degrees. If the angle between the surface normal and the velocity vector are greater than 90 degrees, the applied stress at that point is zero. ISAAC II further assumes that there is an additional normal stress generated due to the rotation of the projectile. This additional force Eqn. (17) is only experienced on the surface of projectile that is rotating into the media. The projectile pictured is moving from left to right. If it is rotating clockwise an additional stress is applied to the bottom half of the projectile as shown in Figure 11.

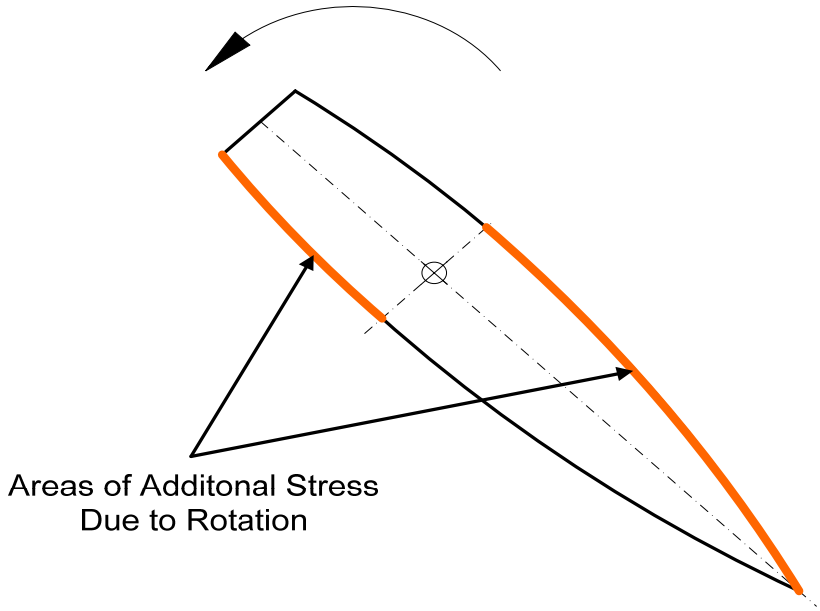


Figure 11. Additional Areas of Stress Due to Projectile Rotation[18]

[18] I M.D. Adley, B.P. Berger, and D.C. Creighton, (1994). "Two-Dimensional Projectile Penetration Into Curvilinear Geologic/Structured Targets: User's Guide for PENCURV-PC," V1.5, Instruction Report SL-94-1 US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 13.

Conversely, if the projectile is rotating counter-clockwise, the additional stress is applied to the top half of the weapon. The total normal stress generated by the ISAAC II formulation is given in Eqn. (19). From these stress formulations, the stress at any point on the surface of the projectile can be determined.

If the stress that an object experiences and the area over which it is applied are known, the force resulting force is given by Eqn. (20).

$$F = \int \sigma dA \tag{20}$$

Previously, this equation was not numerically integrated to solve for the forces acting on the projectile. Instead, it was approximated by the summation of forces acting on a series of differential areas on the surface of the projectile Eqn. (3). The resulting force vector is then used to calculate moments. These forces and moments are then used to solve the equations of motion.

By careful manipulation the PENCURVE stress formulation Eqn. (15) it can be shown that the stress acting on any point on the surface of the projectile is really a function of the axial coordinate of the projectile so long as the curve that defines the shape of the projectile is continuous and continuously differentiable. Since *f(x)* and *g(x)* are polynomials, they meet this requirement. The necessary manipulation of Eqn. (15) will be carried out below.

Each variable in Eqn. (15) is either a constant or a function of the axial coordinate of the projectile. The following discussion is for *f(x)* only, but a similar argument can be for *g(x)*. Starting with projectile radius, $r_p$, $r_p$ is simply the value of *f(x)* evaluated at x. Velocity at a point on the surface of the weapon, *v*, is given by Eqn. (21)

$$\left\|\vec{v}\right\| = \left\|\vec{V}_{CG} + \vec{r} \times \dot{\theta}\right\| \tag{21}$$

where r is the vector going from the CG to that point on the surface of the weapon. If the CG is located at the origin of the body centered coordinate system, the vector r is given by Eqn. (22).

$$r = \begin{bmatrix} x \\ f(x) \\ 0 \end{bmatrix} \tag{22}$$

Note that $\vec{r}$ is solely a function of the axial coordinate of the weapon.

The projection of $\vec{v}$ onto the outward normal, $\vec{N}$, at a point on the weapon is $\vec{v}_n$. The magnitude of $\vec{v}_n$ is $v_n$. If line $\overrightarrow{AB}$ has a slope equal to *m*, a line perpendicular to $\overrightarrow{AB}$ has a slope given by Eqn. (23).

$$m_{\perp} = -\frac{1}{m} \tag{23}$$

The slope of the curve *f(x)* at any point *x* is simply the derivative of *f(x)* evaluated at *x*. Using the definition of the slope of a line, the outward normal at any point is given by Eqn. (24)

$$N = \begin{bmatrix} -m \\ 1 \\ 0 \end{bmatrix} \tag{24}$$

where $m$ is the slope of the curve at point x.

Finally, Z, the depth of any individual point on the surface of the weapon, can be approximated by Eqn. (25) for points forward of the CG and Eqn. (26) for points aft of the CG.

$$Z = Y_{cg} + \sin\theta \tag{25}$$

$$Z = Y_{cg} - \sin\theta \tag{26}$$

where $\theta$ is the incidence angle.

At this point, all quantities in the stress formulation are either constants or functions of the axial coordinate of the weapon.

The area over which the stress is applied to is the surface of the weapon. Since the polynomials that define the surfaces of the weapon, *f(x)* for the top half of the weapon and *g(x)* for the lower half of the weapon, the surface area of the top and bottom halves of the weapon are given by Eqns. (27) and (28)

$$A_{top} = \pi \int_a^b f(x) \sqrt{1 + \left(\frac{df}{dx}\right)^2}\, dx \tag{27}$$

$$A_{bottom} = \pi \int_a^b g(x) \sqrt{1 + \left(\frac{dg}{dx}\right)^2}\, dx \tag{28}$$

The inward normal vector, in whose direction the resultant force acts, can be defined as Eqns. (29)and (30) for the top and bottom half of the projectile, respectively.

$$\hat{n}_{top} = \frac{1}{\sqrt{1+\left(\frac{df}{dx}\right)^2}}\begin{bmatrix}\left(\frac{df}{dx}\right)\\ -1\end{bmatrix} \tag{29}$$

$$\hat{n}_{bottom} = \frac{1}{\sqrt{1+\left(\frac{dg}{dx}\right)^2}}\begin{bmatrix}-\left(\frac{dg}{dx}\right)\\ 1\end{bmatrix} \tag{30}$$

The end result is that the force on the weapon can be defined by the integral given in Eqn. (31).

$$\vec{F}_{top} = \pi\int_a^b \frac{1}{\sqrt{1+\left(\frac{df}{dx}\right)^2}}\begin{bmatrix}\left(\frac{df}{dx}\right)\\ -1\end{bmatrix}\left(\frac{0.625\mu}{Sr_p}\sqrt{\frac{v_n}{v}}+\frac{5.19v}{S}\sqrt{\beta r_p\frac{v_n}{v}}+\frac{5.9\gamma Z}{S^2}\frac{v_n}{v}\right)f(x)\sqrt{1+\left(\frac{df}{dx}\right)^2}\,dx \tag{31}$$

Eqn. (31) reduces to Eqn. (32).

$$\vec{F}_{top} = \pi\int_a^b \begin{bmatrix}\left(\frac{df}{dx}\right)\\ -1\end{bmatrix}\left(\frac{0.625\mu}{Sr_p}\sqrt{\frac{v_n}{v}}+\frac{5.19v}{S}\sqrt{\beta r_p\frac{v_n}{v}}+\frac{5.9\gamma Z}{S^2}\frac{v_n}{v}\right)f(x)dx \tag{32}$$

A similar expression exists for the bottom half of the weapon. Any expression for force that produces a normal force and can be expressed as a function of the axial coordinate of the weapon can be substituted in to Eqn. (32). Now that the force that acts on the weapon is known, the moment can be determined by Eqn. (33).

$$M_{top} = \begin{bmatrix}x\\ f(x)\\ 0\end{bmatrix}\times\pi\left(\frac{0.625\mu}{Sr_p}\sqrt{\frac{v_n}{v}}+\frac{5.19v}{S}\sqrt{\beta r_p\frac{v_n}{v}}+\frac{5.9\gamma Z}{S^2}\frac{v_n}{v}\right)f(x)\begin{bmatrix}\frac{df}{dx}\\ -1\\ 0\end{bmatrix} \tag{33}$$

Both Eqns. (32)and (33) can be numerically integrated and the results used to solve the equations of motion.

The force and moment equations are integrated at each time step by using adaptive Simpson's Quadrature. Using Simpson's rule, the integral of any function, f(x), from a to b is approximated Eqn. (34).

$$\int_a^b f(x) \approx \frac{h}{3}(f(a) + 4f(^{a+b}\!/_{2}) + f(b) = A \tag{34}$$

The result of Eqn. (34) is then compared to the sum of the Simpson's rule approximation for the left half , Eqn. (35), and right half, Eqn. (36), of the interval a to b.

$$\int_a^m f(x) \approx \frac{h}{3}(f(a) + 4f(^{a+m}\!/_{2}) + f(m) = L \tag{35}$$

$$\int_m^b f(x) \approx \frac{h}{3}(f(m) + 4f(^{m+b}\!/_{2}) + f(b) = R \tag{36}$$

The error is given by Eqn. (37).

$$e = \frac{1}{15}(L + R - A) \tag{37}$$

This error is then compared to the tolerance set by the user, T, times L+R. The error is calculated at each time step in conjunction with the ODE solver.

Once the force and moment for each time step has been determined, they are supplied to an ordinary differential equation solver to solve the equations of motion.

## E. EQUATIONS OF MOTION

Having found the forces and moments acting on the surface of the projectile the equations of motion used to solve for the trajectory of the weapon are given by Eqns. (38) and (39).

$$\ddot{\vec{a}} = \frac{\vec{F}_{top} + \vec{F}_{bottom}}{m} \tag{38}$$

$$\ddot{\vec{\theta}} = \frac{\vec{M}_{top} + \vec{M}_{bottom}}{I} \tag{39}$$

This acceleration and angular acceleration are used by a MATLAB ordinary differential equation solver to solve for the position and orientation of the weapon. Default tolerances for the ODE solver are not used as they are too rigorous to be met and the solver fails to find a solution when using them. The forces and moments acting on

the projectile are initially on the order of $10^6$. As the equation is solved through time, significant numerical error begins to accrue. This numerical error makes it impossible for the ODE solver to solve for the trajectory with the default error tolerances. By decreasing the relative error tolerance by one order of magnitude, from $1x10^{-3}$ to $1x10^{-2}$, and the absolute error tolerance by two orders of magnitude, from $1x10^{-6}$ to $1x10^{-4}$, the solver is able to find a solution.

## F. SOLVER TERMINATION

As the projectile velocity nears zero, the ODE solver must continue to reduce the size or its time step to meet the tolerance levels specified by the user. As a result, to solve the differential equations until the magnitude of the CG velocity equals zero takes a significant amount of time. To reduce the amount of time required to solve the differential equations, it is necessary to select a velocity greater than, but close to, zero that terminates the solver. Rather than choose a set velocity, a kinetic energy criterion is used instead. The solver terminates at a velocity equal to the velocity that corresponds to 0.01% of the initial kinetic energy. 99.99% of the kinetic energy expended was chosen as a trade off between accuracy and integration time.

If the projectile has a positive vertical velocity, it is possible that the projectile will broach the surface. If the projectile broaches, it is also necessary that the solver terminates. In this case, the solver will also terminate if Vcg>0.01%Vcg initial if the vertical position of the CG becomes greater than zero.

## G. INTERFACE LIMITS OF INTEGRATION

As the projectile moves through the boundary between two layers, it is important to be able to determine precisely where the boundary interface lies on the surface of the projectile. Consider the projectile in Figure 12 below whose centerline makes an angle,

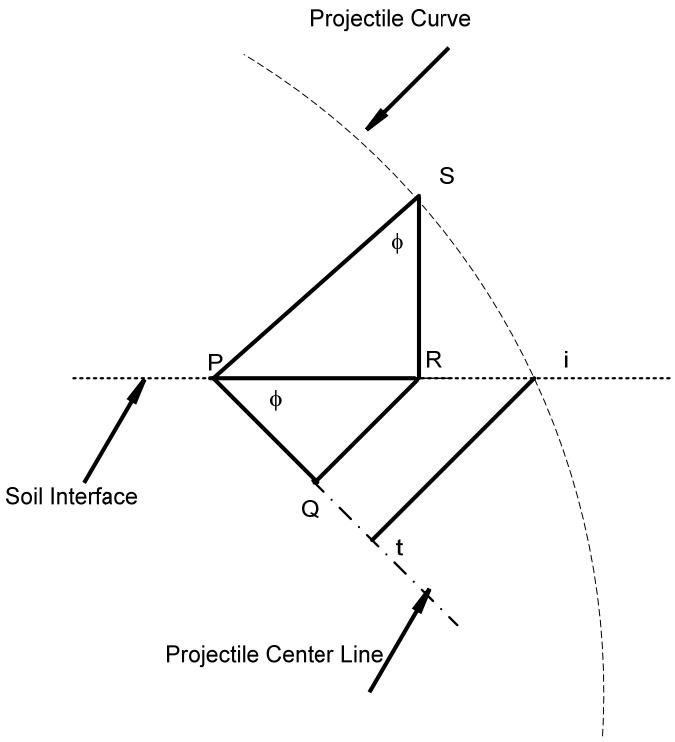


Figure 12. Geometry of the Interface Limits of Integrations

$\phi$, with the surface. The vertical position of the CG is known, either from initial conditions or from the previous time step. With the vertical position of the CG known, the point on the weapon axis that intersects the interface is given by Eqn. (40).

$$p = {Y_{cg}}/{\sin\phi} \tag{40}$$

Point T is defined as the point on the axis of the weapon corresponding to a point on the weapon surface which intersects the interface. This is the lower limit of integration for the top half of the weapon as it moves through the interface. By the angle side angle theorem from geometry, triangles $\triangle PRS$ and $\triangle PQR$ P are similar with $\angle PRS$ and $\angle RPQ$ equal to the incidence angle, $\phi$. The length of $\overrightarrow{PS}$ to *f(x)* evaluated at P. The length of line segment $\overrightarrow{PR}$ is equal to Eqn. (41).

$$\left|\overrightarrow{PR}\right| = \frac{\left|\overrightarrow{PS}\right|}{\sin\theta} \tag{41}$$

The length of line segment $\overrightarrow{PQ}$ is equal to Eqn. (42).

$$\left|\overrightarrow{PQ}\right| = \frac{\left|\overrightarrow{PR}\right|}{\cos\theta} \tag{42}$$

Substituting Eqn. (41) into Eqn. (42) yields an expression for $\left|\overrightarrow{PQ}\right|$ in terms of the axial coordinate of the weapon (Eqn. (43)).

$$\left|\overrightarrow{PQ}\right| = \frac{\left|\overrightarrow{PS}\right|}{\tan\theta} \tag{43}$$

If $\left|\overrightarrow{PQ}\right|$ is used to approximate $\left|\overrightarrow{PT}\right|$, the upper limit of integration for the force acting on the top half of the weapon can be approximated by Eqn. (44).

$$b = \frac{Y_{cg}}{\sin\theta} + \frac{f(P)}{\tan\theta} \tag{44}$$

By a similar argument, the upper limit of integration for the force acting on the bottom half of the weapon can be approximated by Eqn. (45).

$$b = \frac{Y_{cg}}{\sin\theta} - \frac{f(P)}{\tan\theta} \tag{45}$$

## H. WAKE SEPARATION

Experimental observation and three dimensional finite difference modeling have shown that as the projectile travels through the soil, the entire surface of the projectile does not stay in contact with the surrounding soil.[19] This phenomenon has been termed wake separation. The mechanism of wake separation is not fully understood, but does have a significant effect on the total force that is experienced by the projectile. At any point where the soil is not in contact with the surface of the projectile, the stress applied to the projectile at that point is zero. As a result, some method for dealing with wake separation must be devised in order to more closely approximate the forces experienced by the weapon.

The angle formed by the tangent of the projectile surface and the projectile centerline is called the angle of approach, $\phi$. It is assumed that wake separation occurs when the angle of approach is less than some minimum angle. This minimum angle is called the minimum angle of approach, $\phi_{\min}$. If the angle of approach is less than the

[19] R.S. Bernard and D.C. Creighton, (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 13-22.

minimum angle of approach for the wake to maintain contact the wake separates. The minimum angle of approach is generally assumed to be less than 10 degrees.[20] The local angle of approach for any point on the surface of the projectile is defined by Eqn. (46).[21]

$$\phi = \sin^{-1}\left(V_n \big/ V\right) \tag{46}$$

where $V$ is the CG velocity and $V_n$ is the outward component of $V$ normal to the projectile surface.

The minimum angle of approach is given Eqn. (47).[22]

$$\phi_{\min} = \tan^{-1}\left(C \big/ r_o V_z\right) \tag{47}$$

where $r_o$ is the projectile radius where the wake separates and $V_z$ is the vertical velocity of the CG, and $C$ is a constant.

When the wake separates from the projectile, it forms a roughly cone shaped cavity around the weapon. The axis of this cavity, however, does not stay aligned with the axis of the weapon. The axis of the cavity bends away from the centerline of the weapon. The radius of the wake cavity is given by Eqn. (48).[23]

$$r_c = \sqrt{r_o^2 + 2r_o(\xi - \xi_o)\tan\phi_{\min}} \tag{48}$$

where $\xi_o$ is the distance aft of the projectile tip where the wake separates and $\xi$ the point on the projectile's centerline where $r_c$ is being found.

The distance that the centerline of the wake cavity is from the projectile centerline is given by Eqn. (49).[24]

---

[20] R.S. Bernard and D.C. Creighton, (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 14.

[21] Ibid.

[22] Ibid., p. 15.

[23] Ibid.

[24] R.S. Bernard and D.C. Creighton, (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 16.

$$\delta = \frac{1}{2}(\xi - \xi_o)^2 \frac{\dot{\theta}}{V_z} + \xi \sin\left(-\tan^{-1}\left(V_x / V_z\right)\right) \tag{49}$$

where $\dot{\theta}$ is the angular velocity of the projectile about the CG and $V_z$ is the horizontal velocity of the CG.

Empirically derived expressions exist for determining where, if at all, the wake will reattach to the projectile.

The following simplifying assumptions were made about wake separation in this investigation.

- The wake will always be in contact with some portion of both the upper and lower surfaces of the projectile
- The wake does not re-attach aft of the point where the wake detaches.
- The point where the wake detaches is not constrained.

$\phi_{\min}$ is generally assumed to be less than 10 degrees.[25] The wake detaches when $\phi < \phi_{\min}$. $\phi$ equals zero when $V_n$ equals zero. If $\phi_{\min}$ is assumed to be greater than zero, it is guaranteed that the wake will separate when $\phi$=0. In this investigation, it is assumed that the wake is no longer in contact at all points where $V_n \leq 0$.

[25] R.S. Bernard and D.C. Creighton, (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi, p. 14.

# IV. CODE CONSTRUCTION

The overall code process is outlined in Appendix C.

The first task undertaken by the code is to parameterize the shape of the projectile by using a polynomial to define its shape. For all cases, the MK-84 projectile was used. The physical characteristics of the MK-84 bomb are in Table 1. The projectile silhouette is modeled as two tangent ogive curves that meet smoothly at the CG. The equations used to determine the coordinates of the curve are Eqns. (11) and (12) described in PROJECTILE MODEL. Once the coordinates are determined a least squares regression using to determine the degree 3 polynomial that best fits the coordinates. The derivative of this polynomial is also determined. The process is carried out for the top and bottom halves of the weapon.

The user then inputs the speed, initial orientation angle, and SNUM. Next, the user chooses which stress formulation, PENCURV or ISAAC II, to use. Speed and orientation angle are used to determine the initial x and y components of velocity. The initial orientation angle is also used to locate the initial y coordinate of the CG given by Eqns. (50) and (51) .

$$Y_{CG} = L_f \sin\theta \tag{50}$$

$$X_{CG} = L_f \cos\theta \tag{51}$$

where $Y_{CG}$ is the y coordinate of the CG, $X_{CG}$ is the x coordinate of the CG, $L_f$ is the distance from the projectile tip to the CG, and $\theta$ is the incidence angle.

The initial x coordinate of the CG is defined as zero as is the initial angular velocity of the CG. At this point, all initial conditions are defined and all the necessary information exists to solve for the trajectory of the weapon.

The code now chooses between the two stress formulations based on the user inputs. Besides the actual stress formulation used, the code performs all of the same functions. Therefore, only the PENCURV stress formulation case will be discussed.

Additionally, the solving for the forces and moments on the top and bottom halves of the projectile are carried out in the same manner except that different weapon shapes are used, *f(x)* for the top half and *g(x)* for the bottom half.

The ODE solver tolerances and exit termination function are defined. The termination function, EVENTS, calculates the difference between 1% of the initial projectile speed and the current projectile speed. When the projectile has 1% of its initial speed remaining, it has expended 99.9% of its kinetic energy. If the current speed is less than 1% of the initial speed, the ODE solver terminates. The ODE solver calls the either JFINTEGRATE if the ISAAC II stress formulation is used or FINTEGRATE if the PENCURV stress formulation is used.

FINTEGRATE takes time and the state variables of the CG as its inputs. The state variable inputs are assigned variable names for use through out FINTEGRATE and the rest of the code. The transformation matrix UNROT, is defined by Eqn. (52),

$$UNROT = ROT^{T} \tag{52}$$

which is used to transform the results of the force integrations from weapon coordinates to inertial coordinates.

The limits of integration for the stress functions are now determined. If the CG is in the soil half space, the lower limits of integration for both halves of the weapon are defined such that the force is integrated over the entire length of the projectile. If the CG is not in the soil half space, the lower limits of integration are found using the procedure outlined in INTERFACE LIMITS OF INTEGRATION. The upper limit of integration, in all cases is the tip of the projectile.

Using the determined limits of integration, the code now uses adaptive Simpson Quadrature to determine the forces and moments acting on the top and bottom half of the weapon. The stress functions, TALLSTRESS and BALLSTRESS, take as an input a location x along the axis of the weapon. X is in weapon based coordinates. Through the use of global variables, the stress functions have access to the state variables, SNUM, the weapon shape and its derivatives. The stress function begins by determining the outward normal. If *m* is the slope of a line AB, the slope of a line normal to AB is given by Eqn. (23).

The slope of a line is given by Eqn. (53).

$$m = \frac{\Delta y}{\Delta x} \tag{53}$$

where $\Delta y = y_f - y_0$ and $\Delta x = x_f - x_0$.

If it is assumed that the initial position of the vector that defines the outward normal, $N$, is the origin, then the outward normal vector is given by Eqn. (54) .

$$\vec{N} = \begin{bmatrix} -m \\ 1 \\ 0 \end{bmatrix} \tag{54}$$

$N$ is the normalized by dividing by the magnitude of $N$. (Eqn. (55))

$$\hat{n} = \frac{1}{\sqrt{1+m^2}} \begin{bmatrix} -m \\ 1 \\ 0 \end{bmatrix} \tag{55}$$

The code now performs the transformation of CG velocity in inertial coordinates to the velocity of a point on the surface of the weapon in weapon based coordinates. The process begins by forming the transformation matrix $T$ (Eqn. (10)). The CG velocity is then transformed to weapon based coordinates by Eqn. (56).

$$\vec{V}_{CGw} = T\vec{V}_{CGI} \tag{56}$$

If $\vec{r}$ is defined as the vector from the weapon CG to the point on the weapon $P$ define by Eqn. (57) it follows that the vector from the origin to that point is defined by Eqn. (22).

$$P = (x \quad f(x)) \tag{57}$$

The velocity of any point on the weapon in weapon coordinates is given by Eqn. (58).

$$\vec{V}_x = \vec{V}_{CGw} + \vec{r} \times \dot{\theta} \tag{58}$$

Before the forces and moments can be calculated, it must be determined whether or not the wake has separated at the $x$. If the wake has separated at $x$, then no force exists on the surface of the weapon at that point. Previously, it was stated that the wake is no longer in contact at all points where $V_n \le 0$. If $V_n \le 0$ then the dot product of the outward surface normal $\hat{n}$ and $\vec{V}_{CGw}$ is less than or equal to zero. The stress functions calculate

this dot product. If the dot product is greater than zero, it calculates the forces and moments acting at that point. Otherwise, forces and moments are zero. The force acting at the point *x* is found by integrating Eqn. (59).

$$\vec{F} = \int_a^b \hat{n}\sigma(x)dA \tag{59}$$

where

$$dA = \left( \pi f(x)\sqrt{1+\frac{df}{dx}} \right) dx \tag{60}$$

The moment acting on the surface of the weapon are found by integrating Eqn. (61).

$$\vec{M} = \pi \int_a^b \left( \vec{r} \times \hat{n} \left( \sigma(x) f(x) \sqrt{1+\frac{df}{dx}} \right) \right) dx \tag{61}$$

The stress functions return a column vector to FINTEGRATE whose first two elements are the forces acting in the weapon coordinate x and y direction respectively. The last three elements are the moments acting in the x, y and z direction. FINTEGRATE then uses the *UNROT* matrix to transform the forces and moments back to inertial coordinates. The forces and moments in inertial coordinates are what are used by ODE45 to solve the equations of motion.

# V. RESULTS

The IFL method studied in this investigation is able to produce many of the same results produced by the DAFL method. The IFL method is able calculate the trajectory for different stress equations. It is able to replicate the J-hook phenomena while using the ISAAC II stress formulation. Additionally, the IFL method also produces trajectory reversal as predicted by the ISAAC II stress formulation. Most notably, the IFL method was able to accurately predict the path length traveled by the projectile as predicted by Eqn. (7). This is indicative that the total force acting on the weapon calculated by the IFL method is very close to the force calculated by Young. Actual location of the CG and final orientation angle, however, is a function of how well the moments are applied to the projectile. At this time, moments are not well modeled in the IFL method. This is most like due to simplistic wake separation modeling and an inaccurate projectile model. These causes will be discussed in the ANALYSIS AND DISCUSSION section.

To determine how well the IFL method is able to perform these tasks, the IFL output using the ISAAC II stress formulation will be compared to the following data in reference x:

- Trajectory shape
- Maximum CG depth
- Maximum distance downrange
- Cross over angle prediction
- Time from impact to projectile at rest.
- Penetration depth given by Eqn. (7) versus path length traveled.

For the PENCURV stress formulation, the IFL data will be compared to penetration depth as given by EQN 7. A complete tabulation of data is given in Appendix A. Graphs showing the trajectory of each SNUM, Initial Velocity, and Initial Incidence angle are given in Appendix B.

## A. ISAAC II IFL

### 1. Trajectory Shape

In assessing trajectory shape, several questions must be answered.

- Do the IFL trajectories perform a J-hook as predicted by the ISAAC II DAFL data?
- Does the IFL method predict the curve reversal seen in the DAFL data?
- Is the IFL method able to predict projectile broach and ricochet?

The IFL code does show the J-Hook as predicted by ISAAC II. Almost all the ISAAC II generated trajectories show a significant change in projectile incidence angle. However, the true J-Hook has the projectile actually move back toward the surface. This phenomena is seen in the trajectory produced with initial conditions of 900fps initial velocity, 25 initial incidence angle, and SNUM= 11 when using the ISAAC II stress formulation. The trajectory is shown in Figure 13.

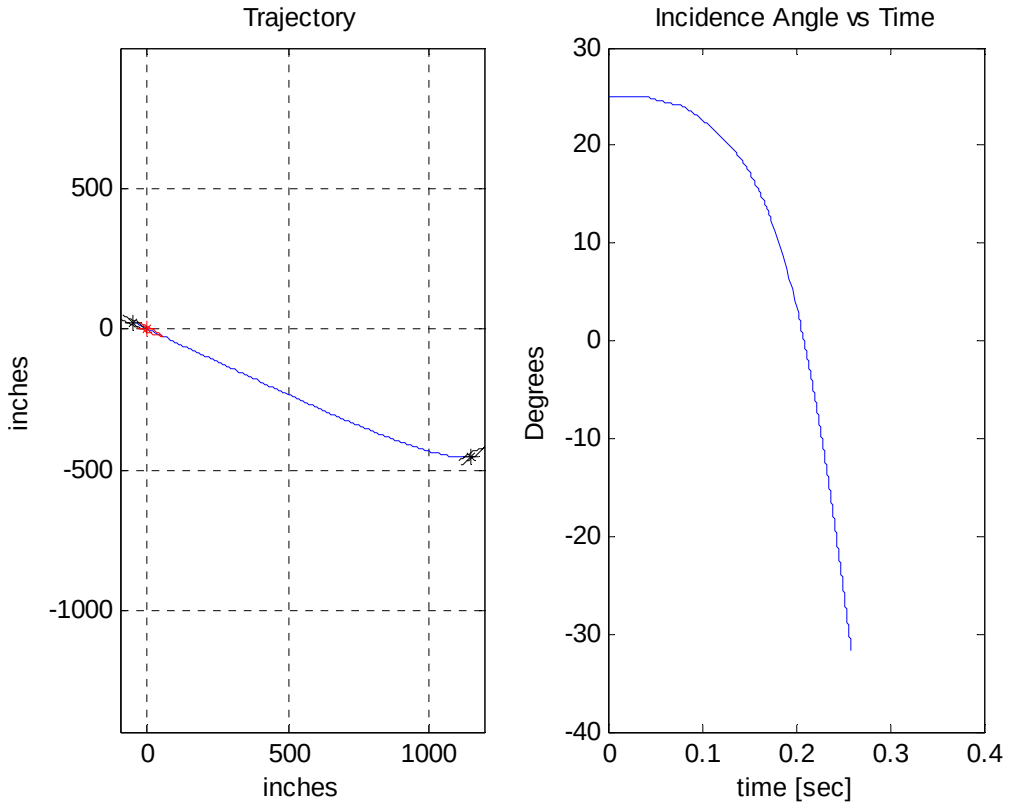


Figure 13. Trajectory for Initial Conditions of 900fps, 25 Degree Incidence Angle, SNUM=11 Using ISAAC II Stress Formulation

A slight upturn is visible on the trajectory plot. The upward motion of the projectile is clearly seen when the y position of the CG is plotted versus time shown in Figure 14.

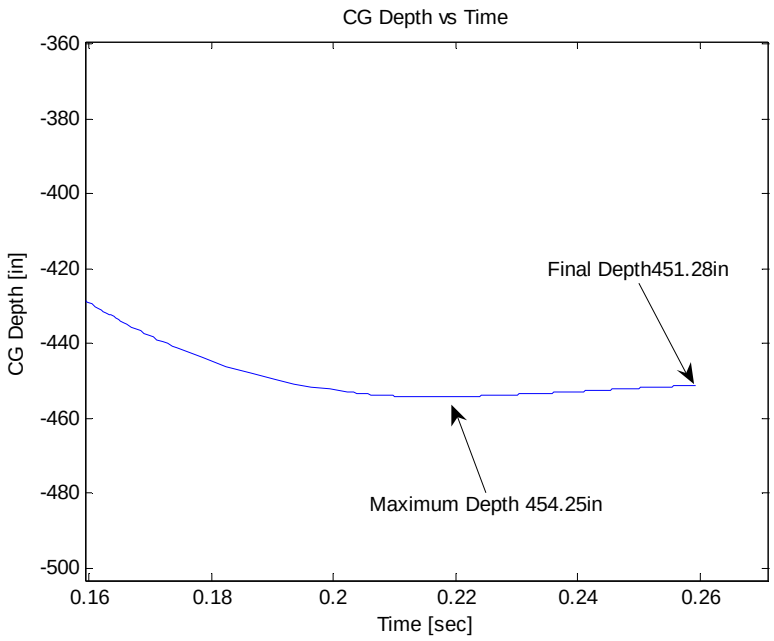


Figure 14. Depth of CG versus Time for Initial Conditions of 900fps, 25 Degree Incidence Angle, SNUM=11 Using ISAAC II Stress Formulation

The difference between the maximum depth and the final depth is small, but it does show that the IFL method is capable of producing a J-Hook trajectory.

### 2. Trajectory Direction Reversal and Cross Over Angle

Trajectory Direction Reversal, where the projectile no longer curves towards the surface but curves away from the surface, is apparent on several IFL Trajectories at all SNUM's and at all initial velocities. These trajectories are shown in Figures 15 and 16. While the curve reversal phenomena can be produced by the IFL method, the angles at which it occurs does not correspond well with the DAFL data. DAFL crossover angles are summarized in Table 3.

| Critical Impact Angle at Which Trajectory Direction-Reversal Occurs | | |
|---|---|---|
| Impact Speed (ft/sec) | SNUM | Critical Angle (Degrees) |
| 300 | 35 | $50 \pm 1$ |
| | 11 | $47.5 \pm 2.5$ |
| | 4 | $40 \pm 5$ |
| 700 | 35 | $65 \pm 5$ |
| | 11 | $66 \pm 1$ |
| | 4 | $65 \pm 5$ |
| 900 | 35 | >70 |
| | 11 | $72.5 \pm 2.5$ |
| | 4 | $65 \pm 5$ |

Table 3. Critical Impact Angle at Which Trajectory Direction Reversal Occurs[26]

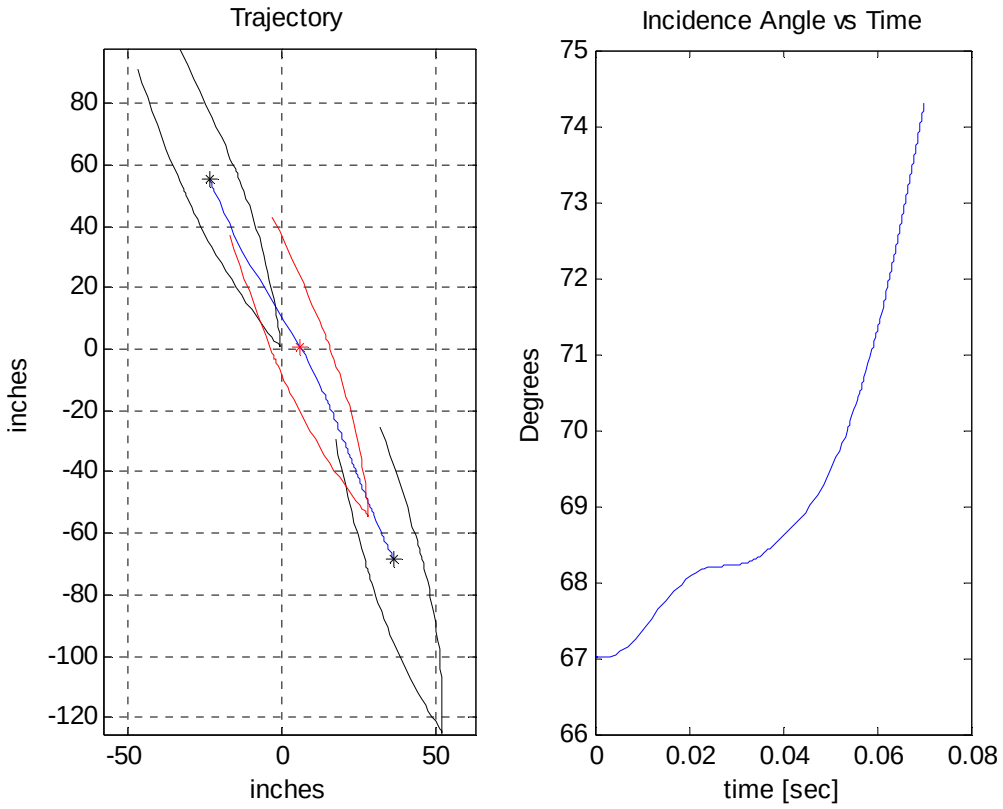


Figure 15. Trajectory Showing Curve Reversal for Initial Conditions of 300fps, 67 Degree Incidence Angle, SNUM=4 Using ISAAC II Stress Formulation

[26] D.C. Creighton, (1988). "Ricochet and Stability of MK82, MK83 and MK84 General Purpose Bombs in Soil," Miscellaneous Paper SL-88-1, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 16.

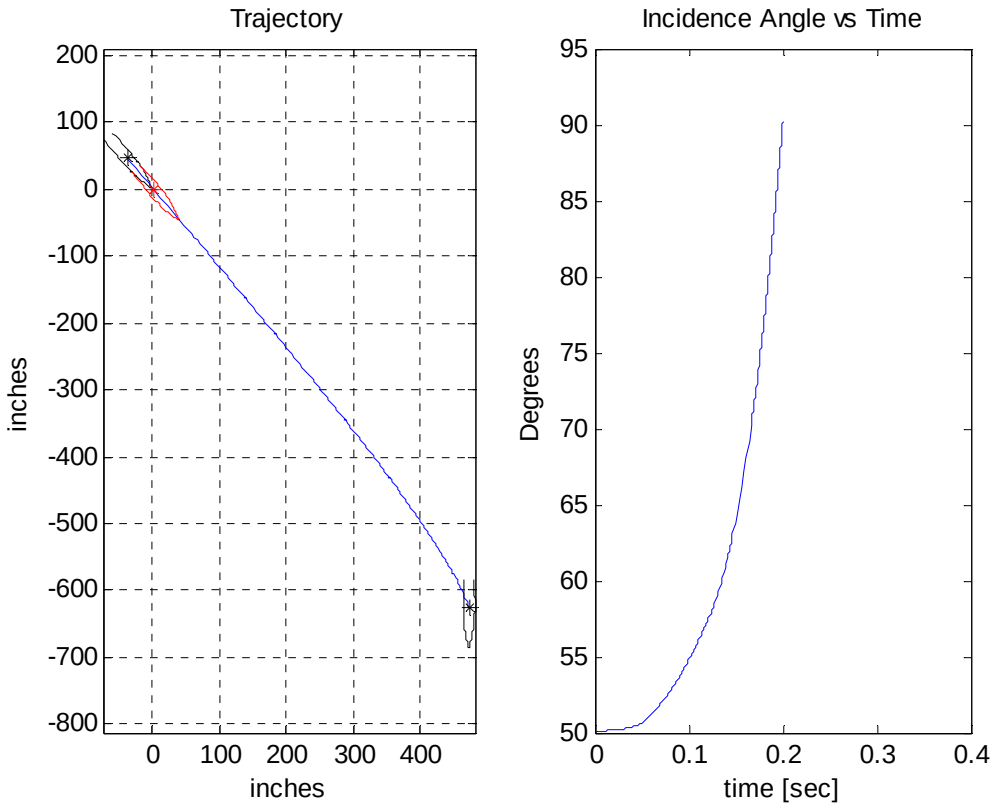


Figure 16. Trajectory Showing Curve Reversal for Initial Conditions of 300fps, 67 Degree Incidence Angle, SNUM=4 Using ISAAC II Stress Formulation

### 3. Cross Over Angle

Cross over angle, the angle greater than the projectile curves away from the surface rather than towards is predicted in the IFL method. Cross over occurs at a discrete angle for all SNUM and initial velocity combinations except for initial velocity of 900fps, incidence angles of 70-80 degrees, and SNUM=11. These trajectories are shown in Figures 17-19.

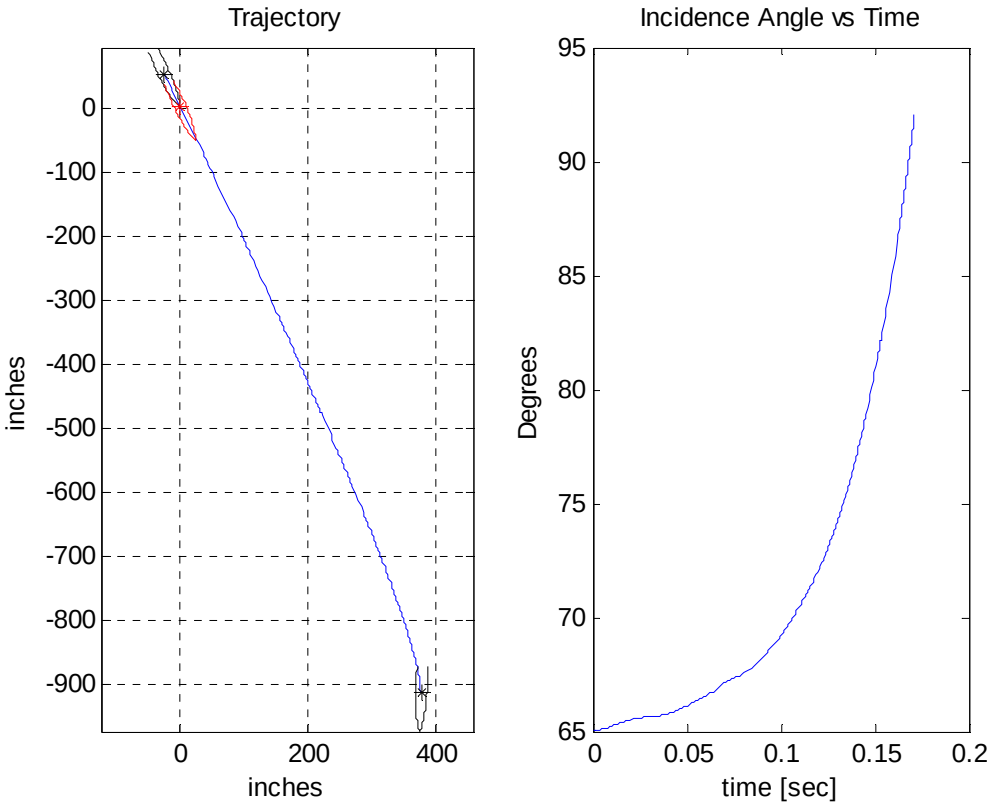


Figure 17. Trajectory Showing for Initial Conditions of 900fps, 70 Degree Incidence Angle, SNUM=11 Using ISAAC II Stress Formulation

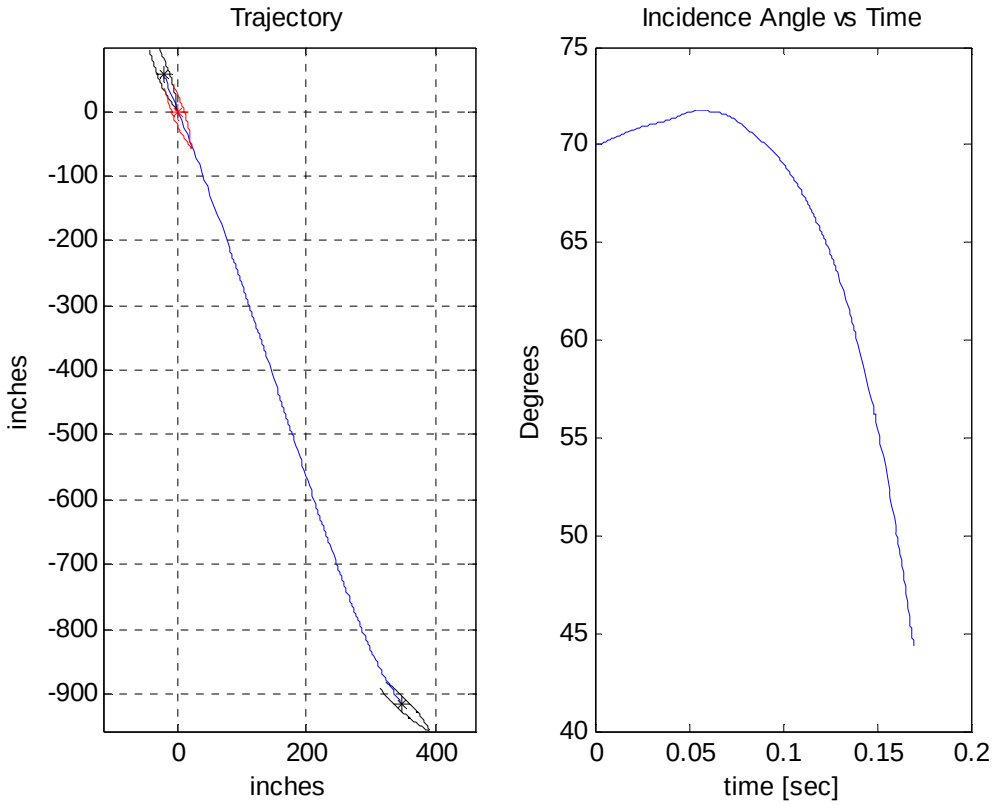


Figure 18. Trajectory Showing for Initial Conditions of 900fps, 75 Degree Incidence Angle, SNUM=11 Using ISAAC II Stress Formulation

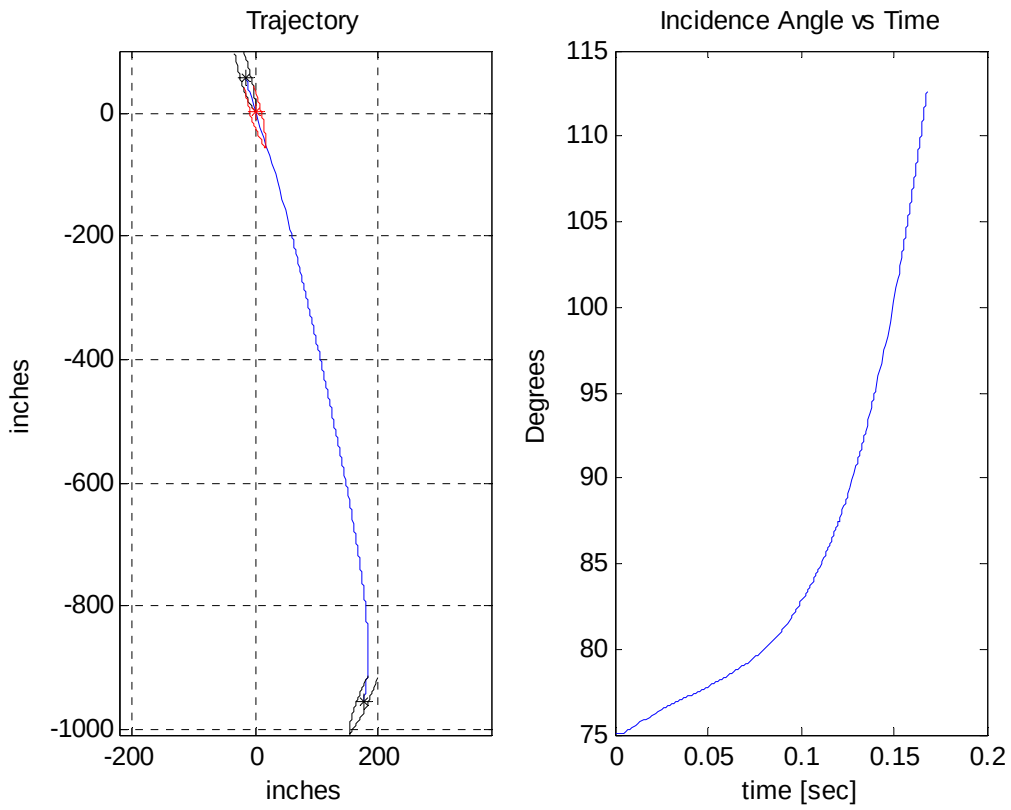


Figure 19. Trajectory Showing for Initial Conditions of 900fps, 80 Degree Incidence Angle, SNUM=11 Using ISAAC II Stress Formulation

The IFL code is not able to predict a ricochet of the projectile or the projectile broaching the surface. The inability to predict ricochet is likely due to not modeling the energy expended in the cratering phase of the weapon impact. The inability to predict broaching likely has many causes including poor wake separation modeling and inaccurate weapon modeling.

#### 4. Maximum CG Depth

When considering maximum CG depth, only DAFL trajectories that did not broach or ricochet were considered. In general, the IFL method was very poor in projecting the maximum CG depth. Percent errors range from as low as 2% to in excess of 200%. A table of all error measurements is given in appendix CC. Error was lowest for low speed, low SNUM high angle of incidence trajectory predictions.

### 5. Maximum Distance Downrange

The IFL method predicted the Maximum Downrange Distance better and more consistently than Maximum CG Depth. For SNUM=4 at both 300fps and 700fps initial velocities and SNUM=11 at 300fps, the error was between 30%-40% for the IFL trajectories that corresponded to penetrating DAFL trajectories. At all other SNUM and initial velocity combinations, error was in excess of 50%.

### 6. Time to Rest

For SNUM=4, the percent error associated with time is less than 20% for all initial velocities and incidence angles. For SNUM=11 and initial velocity of 300fps, two of the three penetrating trajectories have error less than 15%. The other trajectory for 700fps and SNUM=11 penetrating trajectory has an error of 62%. This trajectory also has a very high maximum cg depth error and maximum distance downrange error. Very large error, greater than 100%, exist for all velocities and incidence angles when SNUM=35.

### 7. Path Length

The path length of the IFL trajectory was calculated by approximating the derivative of the trajectory and integrating to find the path length, $L$ (Eqns. 63 and 64).

$$\frac{dy}{dx_i} \approx \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{62}$$

$$L = \int_a^b \sqrt{1 + \left(\frac{dy}{dx_i}\right)^2}\, dx \tag{63}$$

The path length was then compared to the result of Young's Penetration depth equation (Eqn. 7).

For SNUM=4, there very good agreement between path length and Young's Penetration depth for the 300fps and 700fps initial velocity. The average error for 300fps is just over 9% and just less than 15% for 700fps. For SNUM=11, the 300fps and 700fps initial velocities both have error around 15%. For all velocities at SNUM=35 and for an initial velocity of 900fps at both SNUM=4 and SNUM=11 error is greater than 50%.

## B. PENCURVE IFL

### 1. Path Length

The path length for the PENCURV IFL to DAFL comparisons is performed in the same manner as the ISAAC II IFL to DAFL comparison. For SNUM=4 and SNUM=11 for initial velocities of 300fps and 700fps path length errors are less than 30% for all cases. As the angle of incidence increases, the error decreases to a minimum. For SNUM=35 and error is less than 20%. For all other SNUM, incidence angle and initial velocities, error is greater than 60%.

## C. COMPUTING TIME

There is no data available on how much computer time was required to calculate each trajectory using the DAFL method. The longest computing time for any trajectory in this investigation was just over 70 seconds. The shortest computation time was less than two seconds. The average computing time using the ISAAC II stress formulation was 9.7 seconds. The average computing time using the PENCURV stress formulation was 2.0 seconds.

# VI. DISCUSSION AND ANALYSIS

The IFL code presented in this investigation is a successful first step in replacing the DAFL codes currently in use with a more efficient code. The IFL code presented is able to replicate all trajectory phenomena except broach and ricochet. The IFL method also reasonably predicts the path length of the projectile. These two facts indicate that the IFL method is accurately calculating the total force applied to the weapon and to a lesser degree that those forces are being applied in the appropriate locations resulting in the moments that cause rotation. However, based on the inability to accurately predict maximum target depth and maximum downrange distance, there is significant room for improvement in the IFL code. The inability to accurately predict maximum target depth, maximum downrange distance and cross over angle most likely has two fundamental causes. First is the simplistic wake separation model. The other cause is an inaccurate projectile model.

## A. PATH LENGTH AND AVERAGE ACCELERATION

In the development of his SAMPL code, Young makes use of the fact that the penetration distance D, can be related to average force experienced by the weapon. For trajectories employing the ISAAC II stress formulations with initial velocities of 300 and 700 fps at SNUM=4 and SNUM=11 the path length error is less than 15%. For these trajectories the path length is less than the path length predicted by Eqn. (7). For trajectories employing the PENCURV stress formulation all path lengths are less than those predicted by Eqn. (7). Through Eqn. (8), Young relates the penetration distance to the average force. Since the IFL method predicts penetration distances that are less than those predicted by Eqn. (7), the average force that the IFL method calculates must be greater than that experienced by the projectile. Since the errors associated with path length are small, less than 15% for most cases, From this it can be reasoned that the average force the projectile experiences in the IFL method is close to, but greater than, the actual force. Even for the 900fps initial velocity at SNUM=4 and SNUM=11 the IFL path length is much less than the penetration depth predicted by Young. In general, the IFL method applies an average force greater than that predicted by Young. At SNUM less than or equal to 11 for all initial velocities , the average force calculated by the IFL

method is slightly larger than the average force predicted by Young. This is indicative that the stress calculations are correct and that the surface area in contact with the soil is very close to, but larger than that predicted by the DAFL method. This discrepancy in area is either due to poor wake separation modeling or an inaccurate weapon model.

AT SNUM=35, the IFL path length is two to three times greater than the Young predicted penetration depth for the 300fps and 700fps initial velocities. However, for the 900fps initial velocity, the path length error starts at 45% for a twenty degree incidence angle and decreases to an error of just less than 10% for a seventy degree incidence angle. Additionally, at SNUM-11 and SNUM=4 for the 900fps initial velocities the path length is roughly 50% less than the predicted path length for all incidence angles. This data is indicative that the amount of area that is actually in contact with the soil is a not purely a function of weapon geometry as is modeled in this IFL code, but is also most likely a function of initial velocity, and SNUM and possibly incidence angle as well.

Recalling the pertinent wake separation equations (Eqns.(46)-(49)) it is clear that these equations bear out what the IFL code has shown. Wake separation is also a function of projectile speed. The IFL code models wake separation very simplistically. The wake separates when $\phi$=0 as indicated by the dot product of the outward surface normal and the velocity at a point on the surface of the weapon is zero and stays separated from the weapon at all points aft of the separation point. However, the separation angle, $\phi_{\min}$, is a function of the vertical velocity of the CG. $\phi_{\min}$ begins at a minimum at low speed and asymptotically approaches zero with increasing speed. As velocity increases, the wake separates father aft on the weapon, yielding more surface area in contact. The implications on force and moment of the wake separation point moving farther aft are shown in Figures 20-23. This would account for the reasonably close agreement for 900fps at SNUM=35. However, if the wake separates at some $\phi_{\min}$ greater than zero, then less surface area is in contact leading to lower forces applied to the weapon. The overall effect is that the average force would go down and the weapon would have a longer path length. Based on this data, it is likely that a $\phi_{\min}$ greater than zero should be used to model wake separation.

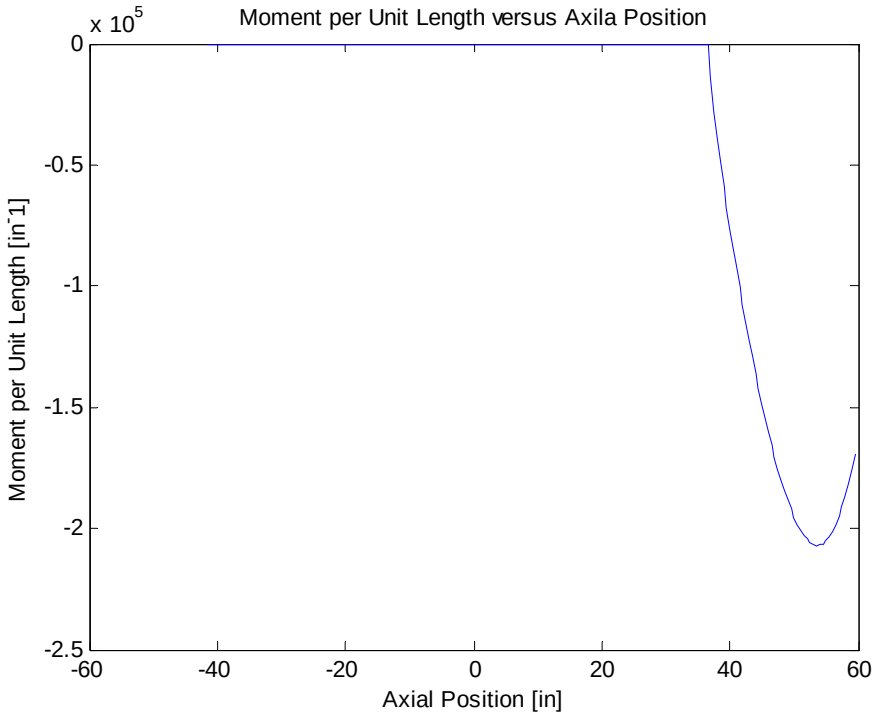


Figure 20. Forces on Top Half of Weapon for 10 Degree Angle of Attack

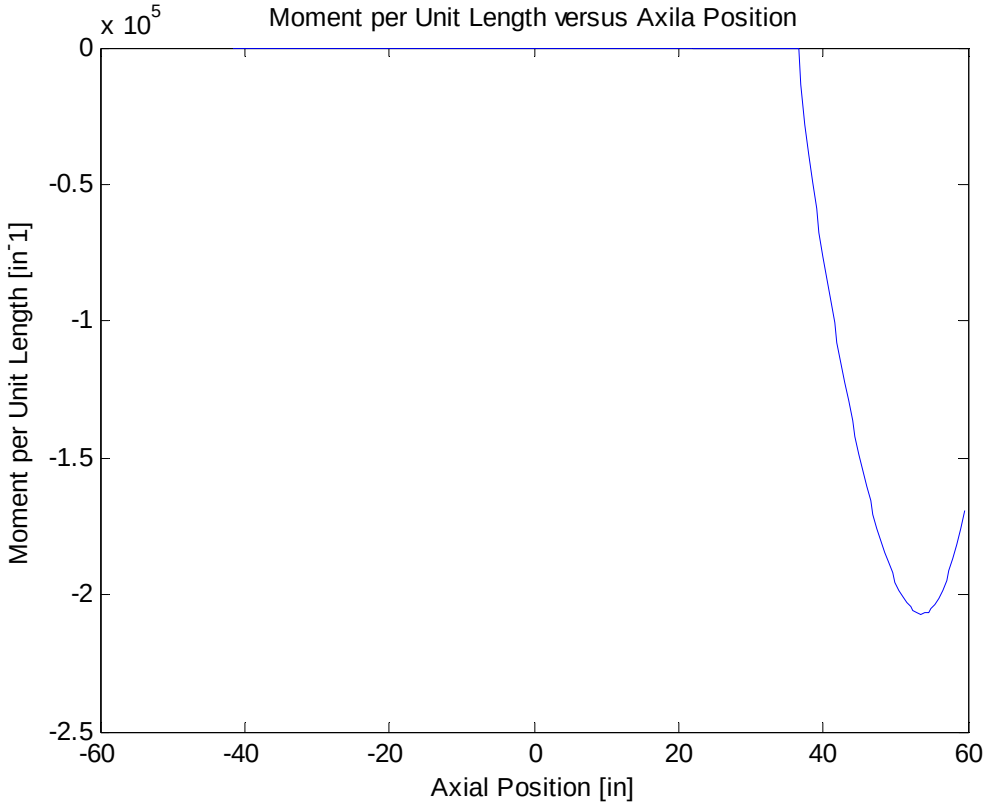


Figure 21. Moment on Top Half of Weapon from 10 Degree Angle of Attack

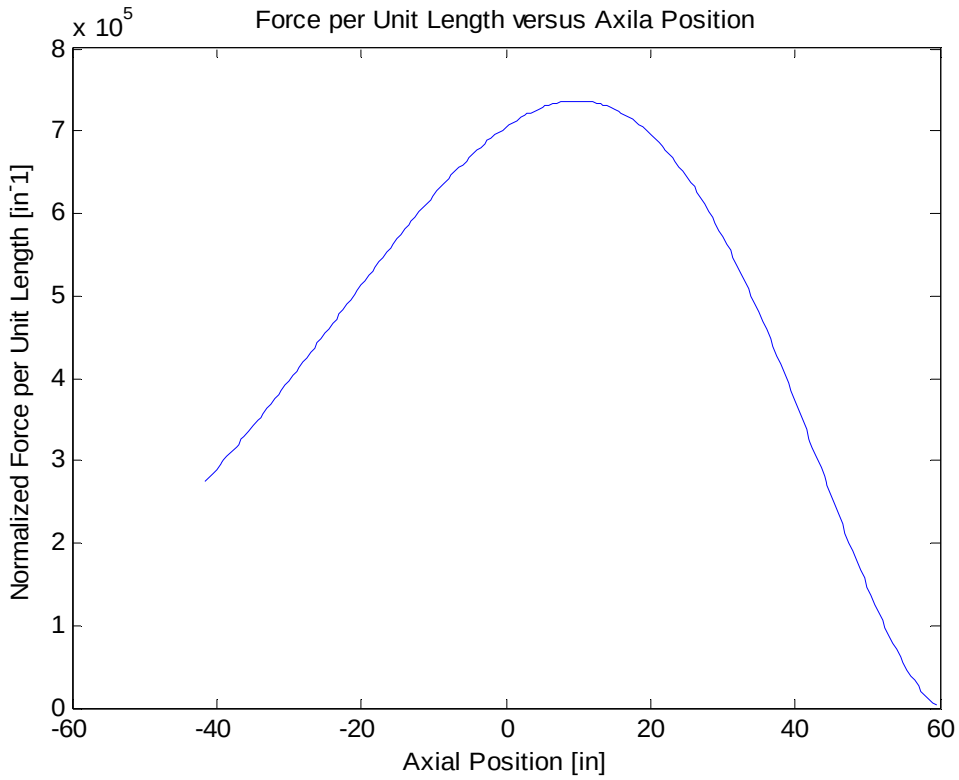


Figure 22. Forces on Bottom Half of Weapon for 10 Degree Angle of Attack

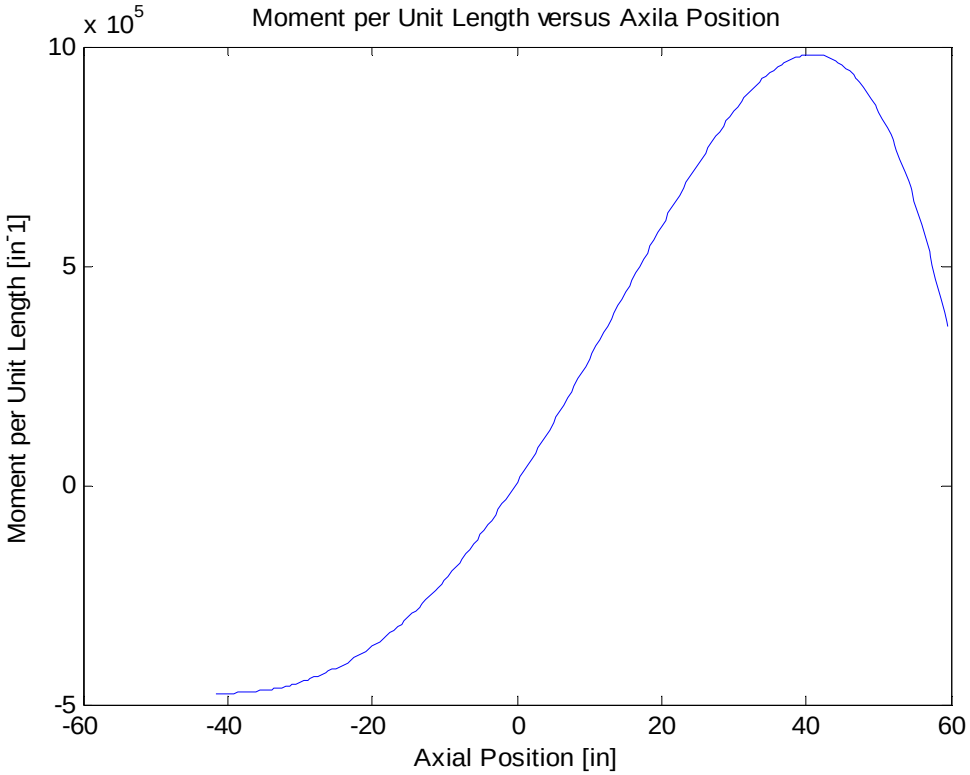


Figure 23. Forces on Bottom Half of Weapon for 10 Degree Angle of Attack

Additional evidence of poor wake separation modeling can be seen by examining the trajectory shapes. While the IFL code is able to predict the J-Hook trajectory it is unable to predict the broach trajectory or accurately predict the cross over angle. In general, the projectile does not rotate fast enough relative to how fast it is slowing down. This is indicative that the moment imbalance between the top and bottom half of the weapon is not large enough.

By examining Figures 20-23, the basis for this can be discerned. Figures 20-23 are plots of forces and moments on the top and bottom half of the projectile for an angle of attack of ten degrees. These figures were generated using the wake separation criteria of $\hat{n} \bullet \vec{V}_x \leq 0$ used in the IFL code. It is apparent that a moment imbalance exists, and that the overall moment will rotate the projectile counter clockwise. However, on the bottom half of the weapon the moment aft of the CG resists the counter clockwise rotation. If the wake separated farther forward on the bottom half of the projectile, a larger counter clockwise moment would exist. This larger moment may be large enough to create the broach trajectory and more accurately model the cross over angle.

The discrepancies in path length can also be accounted for by an inaccurate projectile model. The shape of the projectile is modeled by a degree three polynomial. To calculate the forces acting on the stress is multiplied by the differential area and integrated over the length of the projectile. The differential areas used are represented by Eqns. 28 and 29. These equations are the surface area of a curve rotated about the x axis. The top curve, *f(x)*, is rotated through $\pi$, radians as is the bottom curve to find the surface area of the entire projectile. There is a linear relationship between radius error and surface area error as shown in Figure 24. It is not clear how this error will promulgate as the force acting on the projectile is calculated. However, it is clear that an inaccurate projectile model will lead to errors in surface area which in turn will lead to errors in the total force acting on the projectile.

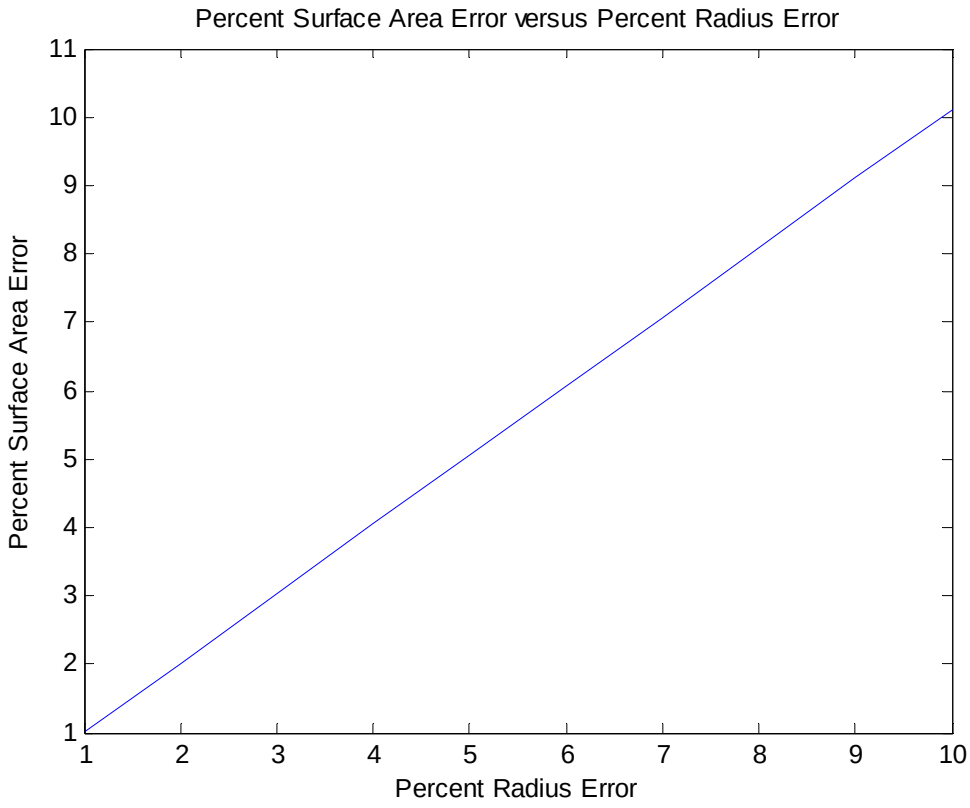


Figure 24. Percent Surface Area Error as a Function of Percent Radius Error

# VII. FURTHER RESEARCH

There are four areas that are in need of additional research in support of refining the IFL method.

- more experimental data,
- better theoretical understanding of wake separation phenomena,
- projectile modeling and
- better modeling of wake separation in the IFL code.

Additional research in any or all of these areas would greatly benefit further development of the IFL code.

Currently, very little experimental data exists for projectile penetration in soil. Additional research would provide many benefits to the IFL development. First, actual trajectory data would be very valuable for code refinement and improvement. If the test projectile could be instrumented this additional data would be of great value. Stress data from the weapon could be compared to IFL stress prediction to not only validate the stress formulation currently in use, but could also be use to locate where the wake separates and be used to refine the wake separation model.

An accurate projectile model is essential to the success of the IFL method since the forces and moments acting on the projectile are a function the projectile shape. If there are errors in the projectile shape, there will be errors in the calculated forces and moments. These force and moment errors will lead to poor trajectory predictions.

Current wake separation theory is refined enough to allow it to be implemented, but the phenomena is not well understood. Research into the fundamental principles of wake separation and improvement in the theoretical model would greatly enhance IFL development as it appears that wake separation plays a large role in determining the trajectory of the weapon. More importantly, however, is making better use of the available wake separation model in the IFL code. The simplistic approach currently employed in the IFL code limits the effectiveness of the code. Improving the wake separation model using existing data in the IFL code should be the first priority of further research. The starting point for improved wake separation modeling would be to increase

the value of $\phi_{min}$ from zero to small finite value less than 10 degrees. Once this is complete, the next logical step would be to model wake reattachment as outlined by Bernard and Creighton (1978).

# VIII. CONCLUSION

The IFL method replaces the large DAFL models utilized by the DAFL method with numerical integration of a stress formulation over a surface of rotation to produce the forces and moments acting on a projectile traveling through the soil. Testing of the effectiveness of this method revealed several things. First, the IFL method is capable of producing a J-hook trajectory using the ISAAC II stress formulation. The IFL method does an adequate job of predicting the total path length of a projectile but fails to accurately predict the shape of the trajectory. This inability to predict the shape of the trajectory is not due to a flaw in the fundamental IFL method, but rather due to a simplistic model of the complicated wake separation phenomena. Overall, the IFL method shows great promise as it is able to re-create the fundamental physical phenomena that occur in a penetration trajectory.

In addition to its ability to reproduce the physical phenomena captured in the DAFL formulations, the IFL method is extremely fast. Single trajectory calculations in a single layer on average take less than 40 seconds to complete. Even if the time per calculation reached one minute, hundreds of runs could be performed in a single day. This represents an immense decrease in computing time currently used to calculate trajectories.

Overall, the IFL method shows great potential to replace the DAFL method of calculating projectile trajectories. The IFL method is able to compute trajectories with reasonable accuracy and with some minor improvements should be able to produce trajectories rivaling DAFL methods in a fraction of the time.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A. IFL TRAJECTORY DATA TABLES

## IFL ISAAC II Trajectories for SNUM=4

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| 300 | 25 | -2.7914 | 0.0772 | 5.9574 | 5.9574 | 19.3124 | 5.656 | 6.5098 |
| | 30 | -3.2861 | 0.0766 | 5.7147 | 5.7147 | 20.7222 | 4.672 | 6.4722 |
| | 35 | -3.6736 | 0.0752 | 5.3135 | 5.3135 | 26.9819 | 4.344 | 6.2561 |
| | 45 | -4.4051 | 0.073994 | 4.625 | 4.625 | 37.681 | 3.9219 | 6.1092 |
| | 50 | -4.762 | 0.072484 | 4.2172 | 4.2172 | 43.51 | 4.75 | 6.1505 |
| | 67 | -5.67 | 0.070151 | 3.028 | 3.028 | 74.29 | 8.2031 | 6.2942 |
| 700 | 20 | -8.5024 | 0.094998 | 24.664 | 24.664 | 0.18242 | 9.4688 | 26.165 |
| | 30 | -12.164 | 0.08516 | 20.793 | 20.793 | 42.493 | 8.6719 | 24.052 |
| | 35 | -13.646 | 0.082356 | 19.062 | 19.062 | 50.282 | 4.6094 | 23.53 |
| | 40 | -14.878 | 0.080468 | 17.517 | 17.517 | 55.209 | 5.3906 | 22.807 |
| | 50 | -16.892 | 0.076127 | 13.874 | 13.874 | 67.293 | 5.2344 | 21.82 |
| | 60 | -18.404 | 0.072668 | 10.496 | 10.496 | 77.032 | 6.5 | 21.007 |
| | 70 | -19.449 | 0.070773 | 7.1986 | 7.1986 | 86.564 | 8.9219 | 20.553 |
| 900 | 20 | -11.493 | 0.098427 | 34.146 | 34.146 | -4.0031 | 12.484 | 36.313 |
| | 30 | -16.173 | 0.087784 | 28.948 | 28.948 | 12.177 | 7.5313 | 32.906 |
| | 35 | -18.835 | 0.084165 | 26.024 | 26.024 | 54.524 | 5.1875 | 32.003 |
| | 40 | -19.855 | 0.080948 | 24.175 | 24.175 | 25.462 | 8.7344 | 30.835 |
| | 50 | -23.057 | 0.075846 | 18.572 | 18.572 | 72.523 | 6.1406 | 29.311 |
| | 60 | -25.199 | 0.073217 | 14.077 | 14.077 | 80.355 | 7.0625 | 28.361 |
| | 70 | -26.41 | 0.06344 | 8.7305 | 8.7305 | 90.947 | 2.9688 | 27.921 |

Table 4. IFL ISAAC II Trajectory Data for SNUM=4

## IFL ISAAC II Trajectories for SNUM=11

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| **300** | 30 | -12.225 | 0.20623 | 21.843 | 21.843 | 13.192 | 7.5313 | 25.045 |
| | 35 | -13.68 | 0.20101 | 20.395 | 20.395 | 13.279 | 6.6875 | 24.413 |
| | 40 | -15 | 0.19534 | 18.579 | 18.579 | 19.302 | 5.5156 | 23.819 |
| | 45 | -16.323 | 0.1902 | 16.818 | 16.818 | 29.632 | 5.7656 | 23.323 |
| | 50 | -17.333 | 0.18672 | 15.193 | 15.193 | 29.982 | 6.3125 | 23.164 |
| **700** | 30 | -33.611 | 0.23777 | 65.839 | 65.839 | -12.949 | 98.141 | 74.273 |
| | 35 | -38.382 | 0.22625 | 60.44 | 60.44 | -2.3188 | 10.953 | 71.623 |
| | 40 | -41.592 | 0.22032 | 55.858 | 55.858 | -2.6645 | 11.281 | 69.443 |
| | 50 | -52.227 | 0.20032 | 39.363 | 39.363 | 90.253 | 9.6875 | 65.51 |
| | 55 | -54.51 | 0.19709 | 34.797 | 34.797 | 92.23 | 11.063 | 64.403 |
| | 60 | -56.349 | 0.19301 | 29.4 | 29.4 | 96.976 | 17.641 | 64.253 |
| | 65 | -57.236 | 0.16839 | 23.317 | 23.317 | 92.192 | 4.0469 | 62.477 |
| | 67 | -58.061 | 0.16827 | 21.257 | 21.257 | 95.059 | 4.4219 | 62.527 |
| | 70 | -58.798 | 0.16712 | 17.72 | 17.687 | 100.39 | 15.797 | 60.228 |
| **900** | 25 | -37.854 | 0.25916 | 95.463 | 95.463 | -31.711 | 15.078 | 103.21 |
| | 30 | -45.262 | 0.24221 | 87.548 | 87.548 | -16.512 | 70.453 | 98.652 |
| | 40 | -55.648 | 0.22293 | 73.782 | 73.782 | -2.9476 | 10.672 | 92.972 |
| | 50 | -63.638 | 0.20912 | 60.68 | 60.68 | 4.8861 | 7.2969 | 87.625 |
| | 60 | -74.071 | 0.17344 | 37.623 | 37.623 | 90.891 | 4.2031 | 83.689 |
| | 65 | -76.145 | 0.17124 | 31.309 | 31.309 | 92.039 | 4.3906 | 82.961 |
| | 70 | -76.295 | 0.16956 | 28.987 | 28.987 | 44.37 | 4.7188 | 81.596 |
| | 75 | -79.459 | 0.16783 | 15.272 | 14.555 | 112.56 | 4.1094 | 65.923 |
| | 80 | -80 | 0.16701 | 7.9924 | 5.8115 | 122.84 | 7.5625 | 34.071 |

Table 5. IFL ISAAC II Trajectory Data for SNUM=11

## IFL ISAAC II Trajectories for SNUM=35

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| **300** | 25 | -33.243 | 0.59591 | 83.239 | 83.239 | -2.9449 | 5.6563 | 89.641 |
| | 30 | -39.127 | 0.57089 | 77.482 | 77.482 | -2.5247 | 6.5781 | 86.919 |
| | 40 | -49.459 | 0.53391 | 65.605 | 65.605 | 10.739 | 7.8281 | 82.149 |
| | 45 | -53.046 | 0.52501 | 61.01 | 61.01 | 6.2577 | 4.5313 | 81.177 |
| | 50 | -58.398 | 0.51128 | 54.013 | 54.013 | 18.722 | 9.2031 | 79.199 |
| | 55 | -62.267 | 0.49876 | 46.973 | 46.973 | 27.279 | 9.6563 | 78.605 |
| **700** | 20 | -70.12 | 0.7572 | 255.77 | 255.77 | -36.16 | 7.5156 | 266.79 |
| | 25 | -109.17 | 0.68532 | 224.84 | 224.84 | 55.936 | 7.1563 | 251.26 |
| | 30 | -103.02 | 0.66866 | 219.61 | 219.61 | -11.847 | 6.4844 | 244.18 |
| | 50 | -154.38 | 0.57388 | 150.55 | 150.55 | 8.5046 | 5.75 | 216.73 |
| | 60 | -168.69 | 0.54803 | 119.25 | 119.25 | 23.39 | 4.3594 | 207.7 |
| | 70 | -185.29 | 0.52966 | 81.373 | 81.373 | 31.758 | 4.5938 | 203.72 |
| **900** | 20 | -91.965 | 0.78423 | 340.93 | 340.93 | -29.396 | 8.3594 | 355.02 |
| | 25 | -157.34 | 0.68459 | 285.33 | 285.33 | 60.633 | 8.0625 | 328.04 |
| | 30 | -132.75 | 0.68706 | 291.01 | 291.01 | -24.839 | 8.25 | 323.08 |
| | 45 | -186.71 | 0.60293 | 222.43 | 222.43 | 2.5961 | 5.3594 | 293.1 |
| | 60 | -219.56 | 0.55932 | 161.12 | 161.12 | 16.625 | 4.5 | 274.89 |
| | 70 | -240.65 | 0.53765 | 111.69 | 111.69 | 27.885 | 4.6406 | 267.64 |
| | 80 | -257.77 | 0.52361 | 25.562 | 12.116 | 150.43 | 10.328 | 72.777 |

Table 6. IFL ISAAC II Trajectory Data for SNUM=4

## IFL PENCRV Trajectories for SNUM=4

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| 300 | 20 | -1.67 | 0.065274 | 4.9241 | 4.9241 | 19.838 | 1.8281 | 4.8713 |
| | 45 | -3.4134 | 0.062262 | 3.7885 | 3.7885 | 44.842 | 1.5313 | 4.8331 |
| | 70 | -5.2799 | 0.062164 | 2.5428 | 2.5428 | 69.974 | 1.4063 | 5.6116 |
| | 80 | -7.3563 | 0.073827 | 2.4026 | 2.4026 | 79.928 | 2.3125 | 7.6142 |
| 700 | 20 | -5.9142 | 0.084206 | 16.568 | 16.568 | 19.455 | 2.0938 | 17.406 |
| | 45 | -11.398 | 0.072735 | 11.698 | 11.698 | 44.605 | 2.2031 | 16.085 |
| | 70 | -15.233 | 0.06815 | 6.0535 | 6.0535 | 69.803 | 2.1563 | 16.36 |
| | 80 | -17.345 | 0.06795 | 4.0071 | 4.0071 | 79.822 | 2.1875 | 17.814 |
| 900 | 20 | -7.9933 | 0.085396 | 22.308 | 22.308 | 19.296 | 2.0156 | 23.493 |
| | 45 | -15.301 | 0.072699 | 15.589 | 15.589 | 44.498 | 2.3281 | 21.75 |
| | 70 | -20.127 | 0.067682 | 7.8037 | 7.8037 | 69.718 | 2.3281 | 21.479 |
| | 80 | -22.377 | 0.067226 | 4.873 | 4.873 | 79.779 | 2.375 | 22.844 |

Table 7. IFL PENCRV Trajectory Data for SNUM=4

## IFL PENCRV Trajectories for SNUM=11

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| **300** | 20 | -6.2227 | 0.19433 | 17.396 | 17.396 | 19.507 | 1.9375 | 18.269 |
| | 45 | -12.21 | 0.1748 | 12.497 | 12.497 | 44.618 | 1.9688 | 17.352 |
| | 70 | -16.403 | 0.16663 | 6.4565 | 6.4565 | 69.802 | 2.2031 | 17.467 |
| | 80 | -18.644 | 0.16745 | 4.2433 | 4.2433 | 79.819 | 2.3594 | 19.006 |
| **700** | 20 | -17.446 | 0.22051 | 48.642 | 48.642 | 18.628 | 1.9063 | 51.541 |
| | 45 | -33.479 | 0.18778 | 33.815 | 33.815 | 44.016 | 2.2188 | 47.341 |
| | 70 | -43.216 | 0.1745 | 16.178 | 16.178 | 69.265 | 2.1094 | 45.885 |
| | 80 | -46.18 | 0.17273 | 8.8966 | 8.8966 | 79.617 | 2.25 | 46.951 |
| **900** | 20 | -23.048 | 0.22626 | 64.447 | 64.447 | 18.161 | 2 | 68.4 |
| | 45 | -44.125 | 0.1904 | 44.573 | 44.573 | 43.675 | 2.1406 | 62.469 |
| | 70 | -56.688 | 0.17614 | 21.128 | 21.128 | 68.995 | 2.1094 | 60.54 |
| | 80 | -60.124 | 0.17405 | 11.32 | 11.32 | 79.385 | 2.0781 | 61.408 |

Table 8. IFL PENCRV Trajectory Data for SNUM=11

## IFL PENCRV Trajectories for SNUM=35

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | CPU Time (sec) | Path Length (feet) |
|---|---|---|---|---|---|---|---|---|
| 300 | 20 | -21.347 | 0.59046 | 59.524 | 59.524 | 18.533 | 1.8594 | 63.193 |
| | 45 | -41.729 | 0.52404 | 42.11 | 42.11 | 43.874 | 2.0625 | 59.059 |
| | 70 | -54.109 | 0.49441 | 20.188 | 20.188 | 69.076 | 2.0625 | 57.594 |
| | 80 | -57.585 | 0.49091 | 10.835 | 10.835 | 79.529 | 2.0781 | 58.694 |
| 700 | 20 | -55.831 | 0.69055 | 159.26 | 159.26 | 15.711 | 1.9219 | 168.81 |
| | 45 | -108.54 | 0.5836 | 110.61 | 110.61 | 41.671 | 2 | 155.09 |
| | 70 | -139.32 | 0.54005 | 52.187 | 52.187 | 67.122 | 1.9844 | 148.48 |
| | 80 | -146.15 | 0.53304 | 26.886 | 26.886 | 77.798 | 1.9531 | 148.71 |
| 900 | 20 | -72.684 | 0.7128 | 209.99 | 209.99 | 14.271 | 1.9063 | 222.6 |
| | 45 | -141.85 | 0.59544 | 145.44 | 145.44 | 40.564 | 1.9844 | 203.18 |
| | 70 | -182.03 | 0.54881 | 68.703 | 68.703 | 66.102 | 1.9531 | 194.73 |
| | 80 | -190.64 | 0.54116 | 35.359 | 35.359 | 76.901 | 1.9844 | 193.79 |

Table 9. IFL PENCRV Trajectory Data for SNUM=35

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX B. DAFL TRAJECTORY DATA TABLES

## DAFL ISAAC II Trajectories for SNUM=4

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | Trajectory Type (Broach or Penetrate) |
|---|---|---|---|---|---|---|---|
| **300** | 25 | 0.03 | 0.028 | 3.2 | 3.2 | 288 | Broach |
| | 30 | 0.94 | 0.115 | 10.3 | 10.3 | 250 | Penetrate |
| | 35 | 2.32 | 0.094 | 8.1 | 8.1 | 257 | Penetrate |
| | 45 | 4.53 | 0.074 | 4.5 | 4.5 | 221 | Penetrate |
| | 50 | 5.13 | 0.071 | 3.4 | 3.4 | 207 | Penetrate |
| **700** | 30 | 2.71 | 0.027 | 22.1 | 22.1 | 303 | Broach |
| | 35 | 4.01 | 0.035 | 30.6 | 30.6 | 305 | Broach |
| | 40 | 5.72 | 0.049 | 27.1 | 27.1 | 277 | Penetrate |
| | 50 | 10.96 | 0.093 | 21.3 | 21.3 | 256 | Penetrate |
| | 60 | 15.55 | 0.08 | 15.2 | 15.2 | 235 | Penetrate |
| | 70 | 19.88 | 0.072 | 2.6 | 2.2 | 176 | Penetrate |
| **900** | 30 | 3.97 | 0.026 | 27.2 | 27.2 | 308 | Broach |
| | 35 | 5.38 | 0.029 | 32.5 | 32.5 | 310 | Broach |
| | 40 | 6.96 | 0.034 | 41.3 | 41.3 | 294 | Broach |
| | 50 | 11.12 | 0.062 | 31.2 | 31.2 | 272 | Penetrate |
| | 60 | 17.17 | 0.089 | 24.8 | 24.8 | 254 | Penetrate |
| | 70 | 26.93 | 0.073 | 4.2 | 3.4 | 174 | Penetrate |

Table 10. DAFL ISAAC II Trajectory Data for SNUM=4

## DAFL ISAAC II Trajectories for SNUM=11

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | Trajectory Type (Broach or Penetrate) |
|---|---|---|---|---|---|---|---|
| **300** | 30 | 2.5 | 0.058 | 19.9 | 19.9 | -33 | Broach |
| | 35 | 4 | 0.077 | 31.5 | 31.5 | -17 | Broach |
| | 40 | 6.29 | 0.12 | 27.5 | 27.5 | -2 | Penetrate |
| | 45 | 11.34 | 0.223 | 23.1 | 23.1 | 13 | Penetrate |
| | 50 | 20.2 | 0.185 | 10.5 | 10.5 | 73 | Penetrate |
| **700** | 30 | 7.94 | 0.049 | 38.2 | 38.2 | 246 | Broach |
| | 35 | 10.24 | 0.054 | 38.2 | 38.2 | 235 | Broach |
| | 40 | 12.55 | 0.057 | 37.9 | 36.9 | 270 | Broach |
| | 50 | 17.21 | 0.065 | 36.9 | 35.9 | 270 | Broach |
| | 55 | 19.53 | 0.066 | 37 | 36.9 | 264 | Broach |
| | 60 | 21.86 | 0.07 | 39.2 | 39.2 | -88 | Broach |
| | 65 | 24.97 | 0.074 | 46.1 | 46.1 | -76 | Broach |
| | 67 | 41.99 | 0.21 | 7.1 | -19.4 | 171 | Penetrate |
| | 70 | 35.86 | 0.116 | 5.1 | -23.2 | 184 | Penetrate |
| **900** | 25 | 7.67 | 0.042 | 46.8 | 46.8 | 133 | Broach |
| | 30 | 10.42 | 0.047 | 45.3 | 45.3 | 172 | Penetrate |
| | 40 | 16.02 | 0.054 | 42.5 | 42.5 | 201 | Penetrate |
| | 50 | 21.54 | 0.058 | 39.4 | 38.7 | 221 | Penetrate |
| | 60 | 26.86 | 0.063 | 35.7 | 33.7 | 239 | Penetrate |
| | 65 | 29.49 | 0.066 | 33.7 | 31.5 | 248 | Penetrate |
| | 70 | 36.92 | 0.076 | 39.3 | 39.3 | -86 | Penetrate |
| | 75 | 41.17 | 0.078 | 4.4 | -14.4 | 237 | Penetrate |
| | 80 | 39.95 | 0.075 | 2.3 | -16.3 | 245 | Penetrate |

Table 11. DAFL ISAAC II Trajectory Data for SNUM=11

## DAFL ISAAC II Trajectories for SNUM=35

| Impact Velocity (feet/sec) | Incidence Angle (Degrees) | Max CG Depth (feet) | Time (sec) | Max Down Range Distance (feet) | Final Down Range Distance (feet) | Final Orientation Angle (Degrees) | Trajectory Type (Broach or Penetrate) |
|---|---|---|---|---|---|---|---|
| **300** | 25 | 5.58 | 0.99 | 36.7 | 36.7 | 221 | Broach |
| | 30 | 7.88 | 0.113 | 37.9 | 37.9 | 173 | Penetrate |
| | 40 | 12.96 | 0.135 | 36.4 | 36.4 | 202 | Penetrate |
| | 45 | 15.86 | 0.147 | 35.8 | 35.8 | 212 | Penetrate |
| | 50 | 63.77 | 0.601 | 53.8 | 53.8 | 50 | Penetrate |
| | 55 | 34.79 | 0.226 | 11.1 | 5.6 | 253 | Penetrate |
| **700** | 20 | 10.65 | 0.084 | 90.4 | 90.4 | 114 | Broach |
| | 25 | 16.91 | 0.153 | 90.8 | 90.8 | 117 | Penetrate |
| | 30 | 23.15 | 0.163 | 86.5 | 86.5 | 119 | Penetrate |
| | 50 | 46.5 | 0.183 | 69.5 | 69.5 | 137 | Penetrate |
| | 60 | 56.36 | 0.19 | 59.5 | 59.5 | 149 | Penetrate |
| | 70 | 79.26 | 0.248 | 12.7 | 5.5 | -29 | Penetrate |
| **900** | 20 | 15.62 | 0.132 | 118.3 | 118.3 | 135 | Broach |
| | 25 | 23.38 | 0.144 | 111.4 | 111.4 | 133 | Penetrate |
| | 30 | 31.15 | 0.157 | 105.1 | 105.1 | 132 | Penetrate |
| | 45 | 53.42 | 0.186 | 88 | 88 | 134 | Penetrate |
| | 60 | 72.03 | 0.191 | 68.8 | 68.8 | 141 | Penetrate |
| | 70 | 82.95 | 0.191 | 54.9 | 54.9 | 149 | Penetrate |

Table 12. DAFL ISAAC II Trajectory Data for SNUM=4

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX C. CODE FLOW CHARTS

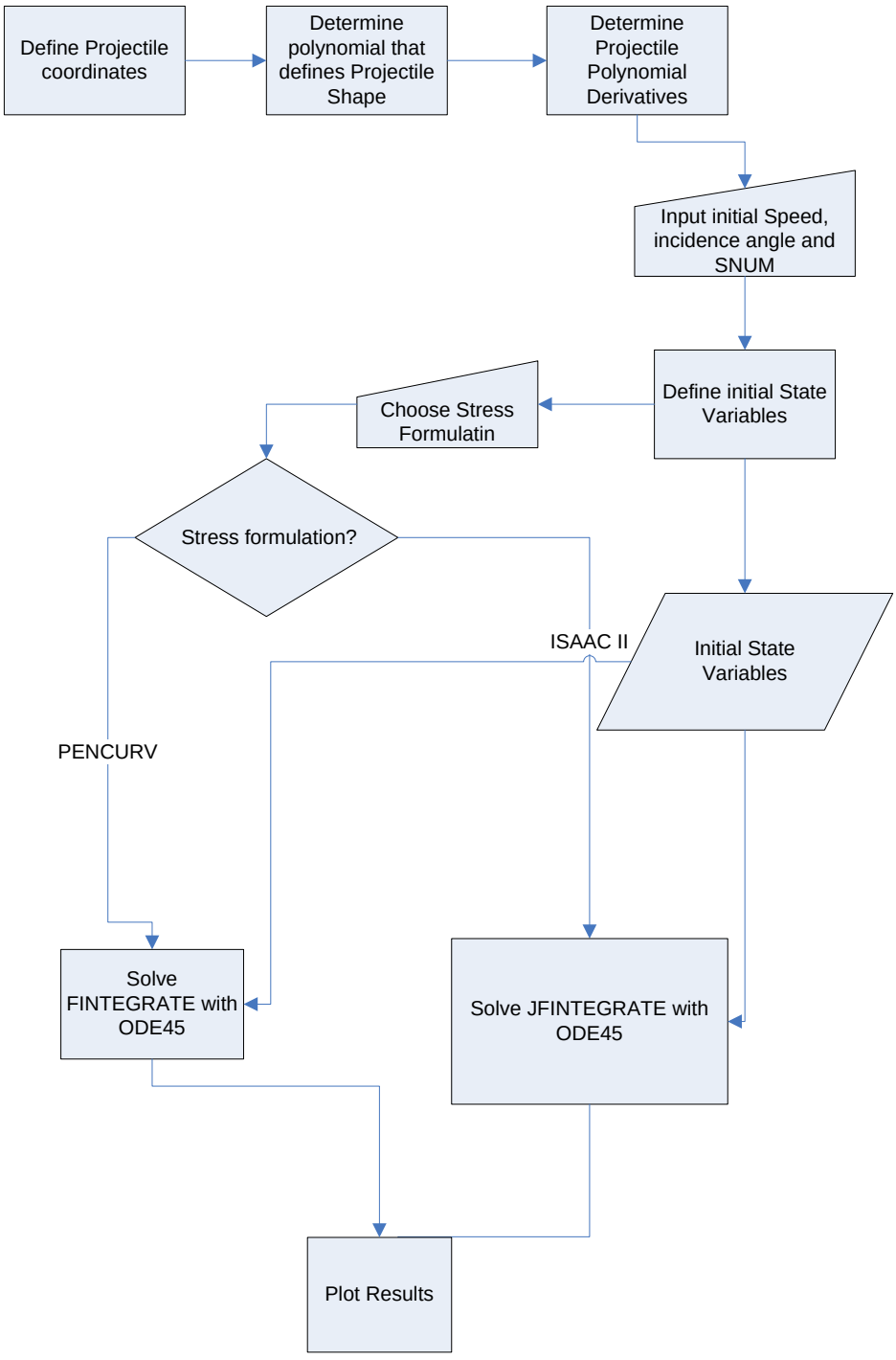


Figure 25. Main Code Flow Chart

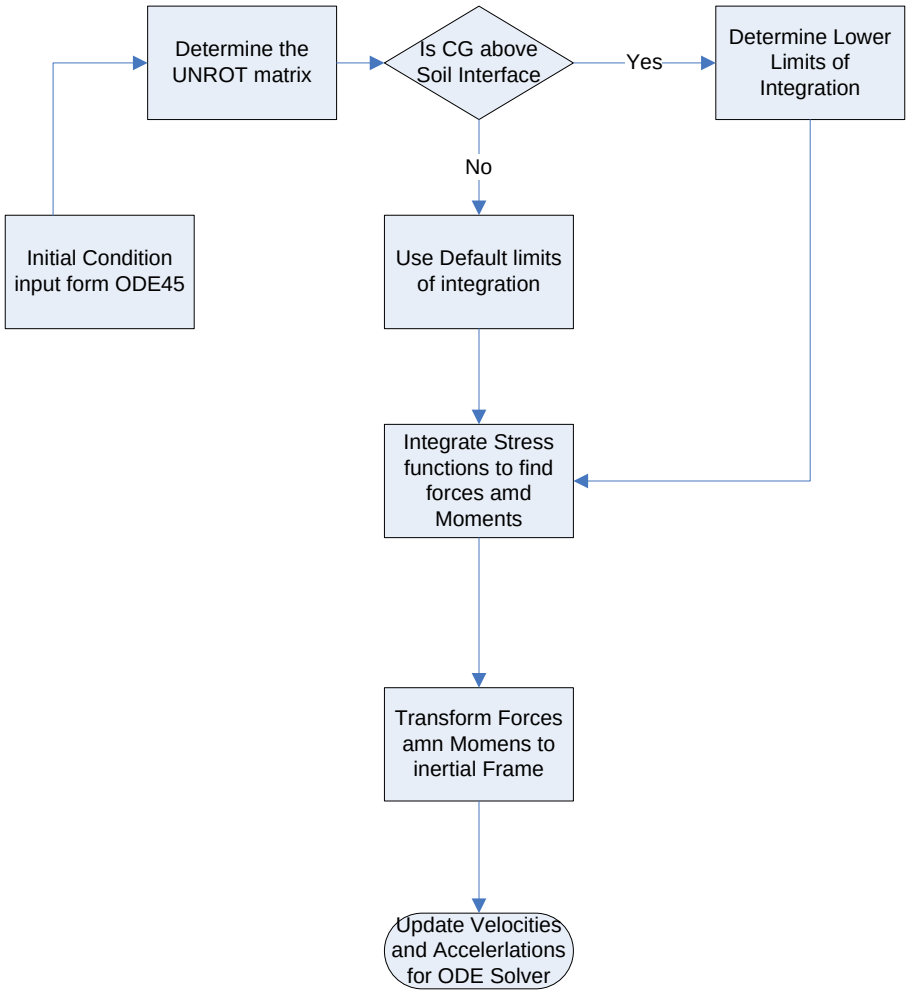


Figure 26. Integrate Code Flow Chart

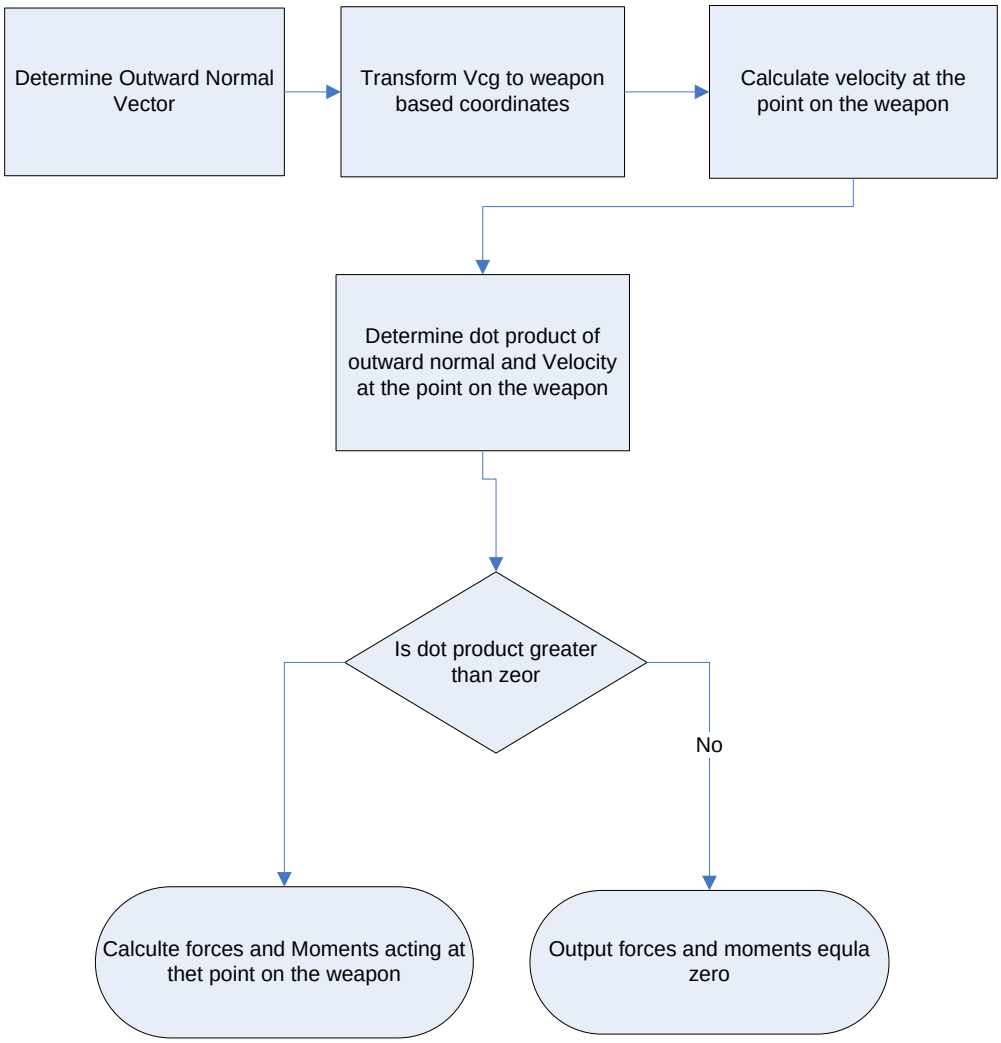


Figure 27. Stress code Flow Chart

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX D. THESIS CODE FOR MAIN THESIS

```
%Main Thesis Code

clc

clear all

close all
%Global Statements

global  VELOCITY  UNROT v SNUM X Y plen

global wpnshape dwpnshape bwpnshape dbwpnshape

global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

%Define Weapon Shape
%Front

a=0;

b=59.5;

Lf = 59.5;

N=30;

n=[1:1:31];

R = 9;

xf=.5.*((b-a).*cos((N+1+.5-n).*pi./(N+1))+a+b);

rho = (R^2 +Lf^2)/(2*R);

uroot =rho^2 -(xf-Lf).^2;

y = sqrt(uroot) +R-rho;

%Back

a=0;

b=106.5;

Lr = 106.5;

xr=.5.*((b-a).*cos((N+1+.5-n).*pi./(N+1))+a+b);
```

```
rho = (R^2 +Lr^2)/(2*R);

uroot =rho^2 -(xr-Lr).^2;

y2 = sqrt(uroot) +R-rho;

X=[xr-Lr xf];

Y=[y2 y(31:-1:1)];

wpnshape=polyfit(X,Y,3);

bwpnshape = -wpnshape;

dwpnshape = polyder(wpnshape);

dbwpnshape = polyder(bwpnshape);

%Input initial Conditions

initalc = input('input initial speed (fps), orientation (angle), and
SNUM');

%Define initial Conditions

theta =initalc(2)*pi/180;

v = initalc(1)*12;

x = 0;

y = 60.*sin(theta);

Vx = v*cos(theta);

Vy = -v*sin(theta);

Thetadot=0;

THETAcg = theta;

SNUM = initalc(3);

INITC = [ x y theta Vx Vy Thetadot];


dot = input('enter "1" for J-Hook modeling ');

if dot == 1

    ATOL = [1e-4 1e-4 1e-4 1e-4 1e-4 1e-4];
```

```
    OPTIONS = odeset('RelTol',1e-2,'AbsTol',ATOL,'Events',@events);
    a=cputime;
    tic

    [t P] = ode45(@jfintegrate,[0,1], INITC,OPTIONS);
    b=cputime;
    toc
    ab=b-a;
else

    OPTIONS = odeset('RelTol',1e-3,'Events',@events);
    a=cputime;
    tic

    [t P] = ode45(@fintegrate,[0,1], INITC,OPTIONS);
    b=cputime;
    toc
    ab=b-a;
end

%Results

speed = ((P(:,4).^2+P(:,5).^2).^.5)./12;

offset = 60.*cos(theta);

q = size(t);

[w,e] = min(abs(P(:,2)));

yyy=diff(P(e(1):q(1),2));

xxx=diff(P(e(1):q(1),1));

der=sqrt((yyy./xxx).^2 + 1);

plen=spline(P(e(1):q(1)-1,1),der);

pathlength = quad(@pl,P(e(1),1),P(q(1),1))/12;

maxx=(max(abs(P(:,1)))-offset)/12;

maxy=min(P(:,2))/12;

qqq=length(t);
t(qqq);
finalangle=P(qqq,3)*180/pi;

DATA = [(P(:,1)-offset)./12 abs(P(:,2)./12) P(:,3)*180/pi t];

shape1 = missileplot(-P(1,3));

shape2 = missileplot(-P(q(1),3));
```

```
shape3 = missileplot(-P(e(1),3));

shape11=shape1(1,:)+P(1,1)-offset;

shape12=shape1(2,:)+P(1,2);

shape13=shape1(3,:)+P(1,1)-offset;

shape14=shape1(4,:)+P(1,2);

shape21=shape2(1,:)+P(q(1),1)-offset;

shape22=shape2(2,:)+P(q(1),2);

shape23=shape2(3,:)+P(q(1),1)-offset;

shape24=shape2(4,:)+P(q(1),2);

shape31=shape3(1,:)+P(e(1),1)-offset;

shape32=shape3(2,:)+P(e(1),2);

shape33=shape3(3,:)+P(e(1),1)-offset;

shape34=shape3(4,:)+P(e(1),2);

%Plot the results
subplot (1,2,1)
%figure(1)

plot(P(:,1)-offset,P(:,2));

hold on

plot(shape11,shape12,'k',shape13,shape14,'k',P(1,1)-offset,P(1,2),'k*')

plot(shape21,shape22,'k',shape23,shape24,'k',P(q(1),1)-
offset,P(q(1),2),'k*')

plot(shape31,shape32,'r',shape33,shape34,'r',P(e(1),1)-
offset,P(e(1),2),'r*')

grid on
k = [initalc pathlength];
axis equal
title('Trajectory')
xlabel('inches')
ylabel('inches')
subplot (1,2,2)
 %figure (2)
%
 plot(t,P(:,3)*180/pi);
```

```
 title('Incidence Angle vs Time')
 xlabel('time [sec]')
ylabel('Degrees')
%
% figure(3)
%
% plot(t,speed)

xcell=[maxy t(qqq) maxx (P(qqq,1)-offset)/12 0 finalangle ab
pathlength]
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX E. FINTEGRATE

```
function [out] = fintegrate (t,INITC)

% Velocity used for finding integration limits
global  VELOCITY  UNROT LIMITS

% Weapon shape global variables
global wpnshape dwpnshape bwpnshape dbwpnshape

%State Variables
global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

%assignments

Ycg = INITC(2);

THETAcg = INITC(3);

xdot = INITC(4);

ydot = INITC(5);

thetadot = INITC(6);

THETADOTcg = INITC(6);

VELOCITY = [INITC(4), INITC(5)];

UNROT =[cos(-THETAcg) -sin(-THETAcg);
       sin(-THETAcg)  cos(-THETAcg) ];

if Ycg<-41*sin(THETAcg)

    lowt = -41.5;

    lowb = -41.5;

else

    p =  Ycg/sin(THETAcg);

    t =p + polyval(wpnshape,p)*tan(THETAcg);

        if t<-41.5

            lowt = -41.5;

        else
```

```
            lowt = t;
        end
     b =p - polyval(wpnshape,p)*tan(THETAcg);
        if b<-41.5
            lowb = -41.5;
        else
            lowb=b;
        end
end
%Find Forces
[TOP] =  adaptsimp(@tallstress,lowt, 59.5,1e-10);
[BOTTOM] = adaptsimp(@ballstress,lowb, 59.5,1e-10);
m=1890/386.4;
j = 1.367e6;
xtot = UNROT*[TOP(1)+BOTTOM(1);TOP(2)+BOTTOM(2)];
%differential equation
dx = xdot;
dy = ydot;
dtheta = thetadot;
dxdot = xtot(1)/m;
dydot = xtot(2)/m;
dthetadot = (TOP(5) +BOTTOM(5))/j;
out = [dx; dy; dtheta; dxdot; dydot; dthetadot];
```

# APPENDIX F. JFINTEGRATE

```
function [out] = jfintegrate (t,INITC)

% Velocity used for finding integration limits
global  VELOCITY  UNROT

% Weapon shape global variables
global wpnshape dwpnshape bwpnshape dbwpnshape

%State Variables
global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

%assignments

Ycg = INITC(2);

THETAcg = INITC(3);

xdot = INITC(4);

ydot = INITC(5);

thetadot = INITC(6);

THETADOTcg = INITC(6);

VELOCITY = [INITC(4), INITC(5)];

UNROT =[cos(-THETAcg) -sin(-THETAcg);
       sin(-THETAcg)  cos(-THETAcg) ];

if Ycg<-41*sin(THETAcg)

    lowt = -41.5;

    lowb = -41.5;

else

    p =  Ycg/sin(THETAcg);

    t =p + polyval(wpnshape,p)*tan(THETAcg);

        if t<-41.5

            lowt = -41.5;

        else
```

```
            lowt = t;
        end
     b =p - polyval(wpnshape,p)*tan(THETAcg);
        if b<-41.5
            lowb = -41.5;
        else
            lowb=b;
        end
end
%Find Forces
[TOP] =  adaptsimp(@jtallstress,lowt, 59.5,1e-10);
[BOTTOM] = adaptsimp(@jballstress,lowb, 59.5,1e-10);
m=1890/386.4;

j = 1.367e6/20;
xtot = UNROT*[TOP(1)+BOTTOM(1);TOP(2)+BOTTOM(2)];
%differential equation
dx = xdot;
dy = ydot;
dtheta = thetadot;
dxdot = xtot(1)/m;
dydot = xtot(2)/m;
dthetadot = (TOP(5) +BOTTOM(5))/j;
out = [dx; dy; dtheta; dxdot; dydot; dthetadot];
```

# APPENDIX G. TALLSTRESS

```
function [forces] = tallstress(x)

%Global Statements

% Velocity used for finding integration limits
global  VELOCITY  SNUM

% Weapon shape global variables
global wpnshape dwpnshape bwpnshape dbwpnshape

%State Variables
global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

m= polyval(dwpnshape,x);

if m~= 0

    normalx =-m;

    normaly = 1;

else

    normaly = 1;

    normalx = 0;

end

scale = 1/(1 +(normalx)^2).^.5;

rot = [cos(THETAcg) -sin(THETAcg) 0
       sin(THETAcg)  cos(THETAcg) 0
       0            0         0];

V = rot*[VELOCITY 0]';

unrot =[cos(-THETAcg) -sin(-THETAcg);
       sin(-THETAcg)  cos(-THETAcg)];

r = [x polyval(wpnshape,x) 0];

tdot = [0 0 THETADOTcg];

Vx = V' + cross(r,tdot);

Vmag = (Vx(1)^2 +Vx(2)^2)^.5;
```

```
Vnmag = scale*(normalx*V(1) +normaly*V(2));%Like the dot product

if Vnmag > 0

    Ratio = (Vnmag/Vmag)^.5;

    R = abs(polyval(wpnshape,x));

    Zi = abs(Ycg);

    ds = pi*polyval(wpnshape,x)*(1 + m^2)^.5;

    n =(1/(m^2 +1)^.5*[m;-1]);

    force = n .*(ds .* Ratio.*(1.023e4 ./ (R.*SNUM) +...
        2.167.*Vmag.*(R.^.5)./SNUM + Ratio.*532.7.*Zi./SNUM.^2));

    moment = (cross([x,polyval(wpnshape,x),0],[force;0]))';

    forces =[force;moment];

else

    forces = [0;0;0;0;0];

end
```

# APPENDIX H. BALLSTRESS

```
      function [forces] = ballstress(x)

%Global Statements

% Velocity used for finding integration limits
global  VELOCITY  SNUM

% Weapon shape global variables
global wpnshape dwpnshape bwpnshape dbwpnshape

%State Variables
global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

m= polyval(dbwpnshape,x);

if m~= 0

    normalx = m;

    normaly =-1;

else

    normaly = -1;

    normalx = 0;

end

scale = 1/(1 +(normalx)^2).^.5;

rot = [cos(THETAcg) -sin(THETAcg) 0
       sin(THETAcg)  cos(THETAcg) 0
       0            0         0];

V = rot*[VELOCITY 0]';

unrot =[cos(-THETAcg) -sin(-THETAcg);
       sin(-THETAcg)  cos(-THETAcg) ];

r = [x polyval(bwpnshape,x) 0];

tdot = [0 0 THETADOTcg];

Vx = V' + cross(r,tdot);

Vmag = (Vx(1)^2 +Vx(2)^2)^.5;
```

```
Vnmag = scale*(normalx*V(1) +normaly*V(2));

if Vnmag > 0

    Ratio = (abs(Vnmag)/Vmag)^.5;

    R = abs(polyval(bwpnshape,x));

    Zi =abs(Ycg);

    ds = pi*abs(polyval(bwpnshape,x))*(1 + m^2)^.5;

    n = 1/(m^2 +1)^.5*[-m;1];

    force = n .*(ds .* Ratio.*(1.023e4 ./ (R.*SNUM) +...
        2.167.*Vmag.*(R.^.5)./SNUM + Ratio.*532.7.*Zi./SNUM.^2));

    %moment = (cross([x-105.92,polyval(bwpnshape,x),0],[force;0]))';
    moment = (cross([x,polyval(bwpnshape,x),0],[force;0]))';
    forces =[force;moment];

else

    forces=[0;0;0;0;0];

end
```

# APPENDIX I. JTALLSTRESS

```
function [forces] = jtallstress(x)

%Global Statements

% Velocity used for finding integration limits
global  VELOCITY  SNUM

% Weapon shape global variables
global wpnshape dwpnshape bwpnshape dbwpnshape

%State Variables
global Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg

M=1890/386.4;

mu=1.26e4;

beta=103;

m= polyval(dwpnshape,x);

if m~= 0

    normalx =-m;

    normaly = 1;

else

    normaly = 1;

    normalx = 0;

end

scale = 1/(1 +(normalx)^2).^.5;

rot = [cos(THETAcg) -sin(THETAcg) 0
       sin(THETAcg)  cos(THETAcg) 0
       0            0         0];

V = rot*[VELOCITY 0]';

unrot =[cos(-THETAcg) -sin(-THETAcg);
       sin(-THETAcg)  cos(-THETAcg)];

D =2.* abs(polyval(wpnshape,x));
```

```
Z = abs(Ycg-x);

angle=(atan(m));

N = 0.508./sqrt(abs(sin(angle)));

r = [x polyval(wpnshape,x) 0];

tdot = [0 0 THETADOTcg];

Vx = V' + cross(r,tdot);

Vnmag = scale*(normalx*V(1) +normaly*V(2));

if Vnmag > 0

    axial = 2./pi .*(mu./(SNUM.*N.*D) + 2.*beta.*Z./(SNUM.^2.*N.^2)
+...
    V(1).*sqrt(beta.*M.*3./7)./(SNUM.*N.*D));

    if Vx(2)>0

        if x>=0

            trans = 4./pi .*(mu./(SNUM.*D) + beta.*Z./(SNUM.^2) +...
            abs(Vx(2)).*sqrt(3.*M.*beta/7)./(SNUM.*D));

        else

            trans = 0;

        end

    else

        trans = 0;

    end

    mag = sqrt(axial.^2+trans.^2);

    n =(1/(m^2 +1)^.5*[m;-1]);

    ds = pi*polyval(wpnshape,x)*(1 + m^2)^.5;

    force=n*mag*ds;

    moment = (cross([x,polyval(wpnshape,x),0],[force;0]))';

    forces =[force;moment];

else
```

```
    forces = [0;0;0;0;0];
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX J. EVENTS

```
function [value,isterminal,direction] = events(t,y)
global v
value = 0.01.*v-(y(4).^2+y(5).^2)^.5; %expended 99.9% of initial KE
isterminal = 1;
direction = 0;
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX K. ADAPTSIMP

From http://www.ima.umn.edu/~arnold/455.f97/programs/adaptsimp.m

```
function int = adaptsimp(f,a,b,tol,lev,fa,fm,fb)
%ADAPTSIMP Adaptive Simpson's rule quadrature
%
%  Call as ADAPTSIMP('f',a,b,tol) to approximate the integral of f(x)
%  over the interval a < x < b, attempting to achieve a relative error
%  of at most tol.  The first argument should be a string containing
%  the name of a function of one variable.  The return value is the
%  approximate integral.
%
%  ADAPTSIMP calls itself recursively with the argument list
%  ADAPTSIMP('f',a,b,tol,lev,fa,fm,fb). The variable lev gives the
%  recursion level (which is used to terminate the program if too many
%  levels are used), and fa, fb, and fm are the values of the integrand
%  at a, b, and (a+b)/2, respectively, (which are used to avoid
%  unnecessary function evaluations).

% initialization, first call only
if nargin == 4
  lev = 1;
  fa = feval(f,a);
  fm = feval(f,(a+b)/2);
  fb = feval(f,b);
end

% recursive calls start here

% start by checking for too many levels of recursion; if so
% don't do any more function evaluations, just use the already
% evaluated points and return
if lev > 10
    disp('10 levels of recursion reached.  Giving up on this
interval.')
    int = (b-a)*(fa+4*fm+fb)/6;
else
% Divide the interval in half and apply Simpson's rule on each half.
% As an error estimate for this double Simpson's rule we use 1/15 times
% the difference between it and the simple Simpson's rule (which is
% an asymptotically exact error estimate).
    h = b - a;
    flm = feval(f,a+h/4);
    frm = feval(f,b-h/4);
    simpl = h*(fa + 4*flm + fm)/12;
    simpr = h*(fm + 4*frm + fb)/12;
    int = simpl + simpr;
    simp = h*(fa+4*fm+fb)/6;
    err = (int-simp)/15;

    % if tolerance is not satisfied, recursively refine approximation
    if abs(err) > tol*abs(int)
       m = (a + b)/2;
```

```
        int = adaptsimp(f,a,m,tol,lev+1,fa,flm,fm) ...
            + adaptsimp(f,m,b,tol,lev+1,fm,frm,fb);
    end
end
```

# APPENDIX L. CODE DOCUMENTATION

## A. MAINTHESIS

| MAINTHESIS | |
|---|---|
| Line Number | Operation |
| 3 | Clears Screen |
| 35 | Clears all variables |
| 7 | Closes all open windows |
| 10 | Defines **VELOCITY UNROT v SNUM X Y plen** as global variables |
| 12 | Defines **wpnshape dwpnshape bwpnshape dbwpnshape** as global variables |
| 14 | Defines **Xcg Ycg Thetacg XDOTcg YDOTcg** and **THETADOTcg** as global variables |
| 19 | Defines **a**=0 as lower limit of nosecone |
| 21 | Defines **b**=59.5 as upper limit of nosecone |
| 23 | Defines **Lf**=59.5. This is the length of the nosecone |
| 25 | Defines **N**=30 |
| 27 | Defines **n** row vector from 1 to 31 |
| 29 | Defines **R**=9. This is the maximum projectile radius. |
| 31 | Sets **xf** equal to the first 31 roots of the Chebyschev polynomial scaled between 0 and 59.5. |
| 33 | Defines **rho** as the radius of the nosecone ogive |
| 35 | Defines **uroot** |
| 37 | Caclulates **y** coordinates of projectile nosecone using **xf** |
| 41 | Defines **a**=0 as lower limit of afterbody |
| 43 | Defines **b**=106.5 as upper limit of afterbody |
| 45 | Defines **Lr**=106.5. This is the length of the nosecone |
| 47 | Sets **xr** equal to the first 31 roots of the Chebyschev polynomial scaled between 0 and 106.5 |
| 49 | Defines **rho** as the radius of the nosecone ogive |
| 51 | Defines **uroot** |
| 53 | Caclulates **y2** coordinates of projectile afterbody using **xr** |
| 55 | Assembles **X** as a vector containing the x coordinates of the afterbody and nosecone |
| 57 | Assembles **Y** as a vector containing the y coordinates of the afterbody and nosecone |
| 59 | Defines **wpnshape** as the degree three polynomial whose coefficients come from a least squares fit of **X** and **Y.** |
| 61 | Defines **bwpnshape=-wpnshape** |
| 63 | Defines **dwpnshape** as the derivative of **wpnshape** |
| 65 | Defines **dbwpnshape** as the derivative of **bwpnshape** |
| 69 | Sets **initialc** equal to a vector containing the initial projectile velocity, |

| MAINTHESIS | |
|---|---|
| Line Number | Operation |
| | incidence angle and SNUM. |
| 73 | Defines **theta** as initial incidence angle in radians |
| 75 | Defines **v** as initial speed in inches per second. |
| 77 | Defines **x**=0 |
| 79 | Defines **y** as initial y coordinate of CG |
| 81 | Defines **Vx** as initial x velocity of CG |
| 83 | Defines **Vy** as initial y velocity of CG |
| 85 | Defines **Thetadot**=0 |
| 87 | Defines **THETAcg=theta** |
| 89 | Defines **SNUM** equal to user inputted SNUM |
| 91 | Assembles **INITC** vector containing initial conditions for ODE solver |
| 95 | Defines **dot** to determine which stress formulation is to be used. |
| 97 | If **dot**=1 |
| 99 | Defines **ATOL=**$1x10^{-4}$ |
| 101 | Sets **OPTIONS** equal to a structure defined by odeset. The function "events" is used to terminate the integration of ode45 when 99.9% of the projectiles initial kinetic energy has been expended. |
| 103 | Starts timer |
| 105 | Solves *jfintegrate* using ODE45 |
| 107 | Stops timer |
| 109 | If **dot** does not equal 1 |
| 111 | Sets **OPTIONS** equal to a structure defined by odeset. The function "events" is used to terminate the integration of ode45 when 99.9% of the projectiles initial kinetic energy has been expended. |
| 113 | Starts timer |
| 115 | Solves *fintegrate* using ODE45 |
| 117 | Stops time |
| 122 | Calculates projectile speed at each time step |
| 124 | Calculates x offset to put tip of projectile initially at the origin of the inertial frame |
| 126 | Defines **q** as the size of time vector **t** |
| 130 | Defines **yyy** as the difference in y coordinate between adjacent time steps |
| 132 | Defines **xxx** as the difference in x coordinate between adjacent time steps |
| 134 | Defines **der** as the path length traveled by projectile between time steps |
| 136 | Defines **plen** as the cubic spline of **der** and the x coordinate of the projectile. |
| 138 | Finds **d** by integrating **plen** |
| 140 | Defines **DATA** as [x position of CG in feet, y position of CG in feet, angle in degrees, time in seconds] |
| 142-170 | Define weapon shapes to be plotted on output plot. |
| 176-184 | Plots the weapon trajectory |
| 195 | Plots incidence angle versus time. |

Table 13. MAINTHESIS Code Documentation

## B. EVENTS

| EVENTS | |
|---|---|
| Line Number | Operation |
| 1 | Defines "events" as a function that takes two inputs, **t** and **y,** and outputs **value, isterminal,** and **direction** |
| 7 | Allows "events" to access global variable **v** |
| 8 | Sets **value** to the difference between the velocity that corresponds to the projectile velocity when it has 0.25% of its initial kinetic energy remaining and the current projectile velocity |
| 9 | Defines **isterminal** as 1 |
| 10 | Defines **direction** as 0 |

Table 14. EVENTS Code Documentation

## C. FINTEGRATE

| FINTEGRATE | |
|---|---|
| 1 | Defines "fintegrate" as a function that takes two inputs, **t** and **INITC,** and outputs **out** |
| 4 | Allows "fintegrate" to access global variables **VELOCITY** and **UNROT** |
| 7 | Allows "fintegrate" to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 10 | Allows "fintegrate" to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 14 | Sets **Ycg** equal to the y position of the CG |
| 16 | Sets **THETAcg** equal to the incidence angle of the CG |
| 18 | Sets **xdot** equal to the velocity in the x direction of the projectile |
| 20 | Sets **ydot** equal to the velocity in the y direction of the projectile |
| 22 | Sets **thetadot** equal to the angular velocity of the projectile |
| 24 | Sets **THETADOTcg** equal to the angular acceleration of the projectile |
| 26 | Sets **VELOCITY** to the x and y velocity of the projectile contained in INITC(4) and INITC(5) respectively |
| 28-29 | Defines **UNROT** as the matrix that transforms from the body centered coordinate system to the inertial coordinate system |
| 31-65 | Determines interface limits of integrations |
| 31 | If **Ycg** is in soil half space, |
| 33 | Sets **lowt**=-41.5 |
| 35 | Sets **lowb**=-41.5 |
| 37 | If **Ycg** is not in soil half space, |
| 39 | Defines **p as** place on weapon axis that intersects soil interface |
| 41 | Approximates point **t** where projectile intersects soil interface |
| 43 | If **t** less than or equal to -41.5 |

| FINTEGRATE | |
|---|---|
| 45 | Sets **lowt**=-41.5 |
| 47 | Else |
| 49 | **lowt**=**t** |
| 51 | end |
| 53 | Approximates point **b** where projectile intersects soil interface |
| 55 | If **b** less than or equal to -41.5 |
| 57 | Sets **lowb**=-41.5 |
| 59 | Else |
| 61 | **lowb**=**b** |
| 63 | end |
| 65 | end |
| 69 | Defines **TOP** as integral of *tallstress* from **lowt** to 59.5 |
| 71 | Defines **TOP** as integral of *ballstress* from **lowb** to 59.5 |
| 73 | Defines **m** as projectile mass |
| 75 | Defines **j** as projectile moment of inertia |
| 77 | Transforms forces from weapon frame to inertial frame |
| 80-92 | Define differential equations |
| 80 | Sets **dx=xdot** |
| 82 | Sets **dy**=**ydot** |
| 84 | Sets **dtheta=thetadot** |
| 86 | Sets **dxdot=xtot(1)/m** |
| 88 | Sets **dydot=xtot(2)/m** |
| 90 | Sets **dthetadot=(TOP(5)+BOTTOM(5))/j** |
| 92 | Sets **out**=[**dx; dy; dtheta; dxdot;dydot;dthetatdot]** |

Table 15. FINTEGRATE Code Documentation

## D. JFINTEGRATE

| JFINTEGRATE | |
|---|---|
| 1 | Defines “fintegrate” as a function that takes two inputs, **t** and **INITC,** and outputs **out** |
| 4 | Allows “fintegrate” to access global variables **VELOCITY** and **UNROT** |
| 7 | Allows “fintegrate” to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 10 | Allows “fintegrate” to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 14 | Sets **Ycg** equal to the y position of the CG |
| 16 | Sets **THETAcg** equal to the incidence angle of the CG |
| 18 | Sets **xdot** equal to the velocity in the x direction of the projectile |
| 20 | Sets **ydot** equal to the velocity in the y direction of the projectile |
| 22 | Sets **thetadot** equal to the angular velocity of the projectile |
| 24 | Sets **THETADOTcg** equal to the angular acceleration of the projectile |

| JFINTEGRATE | |
|---|---|
| 26 | Sets **VELOCITY** to the x and y velocity of the projectile contained in INITC(4) and INITC(5) respectively |
| 28-29 | Defines UNROT as the matrix that transforms from the body centered coordinate system to the inertial coordinate system |
| 31-65 | Determines interface limits of integrations |
| 31 | If **Ycg** is in soil half space, |
| 33 | Sets **lowt**=-41.5 |
| 35 | Sets **lowb**=-41.5 |
| 37 | If **Ycg** is not in soil half space, |
| 39 | Defines **p as** place on weapon axis that intersects soil interface |
| 41 | Approximates point **t** where projectile intersects soil interface |
| 43 | If **t** less than or equal to -41.5 |
| 45 | Sets **lowt**=-41.5 |
| 47 | Else |
| 49 | **lowt**=**t** |
| 51 | end |
| 53 | Approximates point **b** where projectile intersects soil interface |
| 55 | If **b** less than or equal to -41.5 |
| 57 | Sets **lowb**=-41.5 |
| 59 | Else |
| 61 | **lowb**=**b** |
| 63 | end |
| 65 | end |
| 69 | Defines **TOP** as integral of *jtallstress* from **lowt** to 59.5 |
| 71 | Defines **TOP** as integral of *jballstress* from **lowb** to 59.5 |
| 73 | Defines **m** as projectile mass |
| 75 | Defines **j** as projectile moment of inertia divided by 20 |
| 77 | Transforms forces from weapon frame to inertial frame |
| 80-92 | Define differential equations |
| 80 | Sets **dx=xdot** |
| 82 | Sets **dy**=**ydot** |
| 84 | Sets **dtheta=thetadot** |
| 86 | Sets **dxdot=xtot(1)/m** |
| 88 | Sets **dydot=xtot(2)/m** |
| 90 | Sets **dthetadot=(TOP(5)+BOTTOM(5))/j** |
| 92 | Sets **out**=[**dx; dy; dtheta; dxdot;dydot;dthetatdot]** |

Table 16. JFINTEGRATE Code Documentation

## E. JTALLSTRESS

| JTALLSTRESS | |
|---|---|
| Line Number | Operation |
| 1 | Defines "tallstress" as a function that takes one input, **x** and outputs **forces** |
| 6 | Allows "tallstress" to access global variables **VELOCITY** and **SNUM** |
| 9 | Allows "tallstress" to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 16 | Allows "tallstress" to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 18 | Defines **M** as the mass of the projectile in slugs |
| 20 | Sets **m** equal to the slope of the projectile shape at **x** |
| 23 | If **m** does not equal zero |
| 26 | If **m** does not equal zero sets **normalx** to the negative of **m** |
| 27 | If **m** does not equal zero sets **normaly** equal to 1 |
| 28 | else |
| 31 | If **m** does equals zero sets **normaly** to equal to 1 |
| 32 | If **m** does equals zero sets **normalx** equal to 0 |
| 34 | Defines **scale** as the inverse of the length of the vector defined by **normalx** and **normaly** |
| 38-40 | Defines rotation matrix, **rot,** that transforms from inertial coordinates to projectile centered coordinates |
| 42 | Transfroms **V** from inertial coordinates to projectile centered coordinates |
| 43-44 | Defines rotation matrix, **urot,** that transforms from projectile centered coordinates to inertial coordinates. |
| 47 | Defines **D** as the diameter of the projectile at **x** |
| 48 | Defines **Z** as the depth of the projectile surface at **x** |
| 49 | Sets **angle** to be the angle formed by the slope of the tangent line at **x** and the axial axis |
| 50 | Sets **N** equal to the local nose performance coefficient |
| 51 | Defines **mu** as 1.24e4 |
| 52 | Defines **beta** as 103 |
| 53 | Defines **r** as the vector going from CG to the point on the projectile surface defined by **x** and **wpnshape**(**x**). |
| 54 | Sets **tdot** equal to the angular velocity vector |
| 55 | Sets **Vx** equal to the velocity of the surface of the projectile at [ **x wpnshape**(**x**)] |
| 56 | Sets **Vmag** equal to the speed of the projectile surface at [ **x wpnshape**(**x**)] |
| 57 | Sets **Vnmag** equal to the dot product of the normal surface vector and the velocity on the surface of the projectile |
| 60 | If **Vnmag** is greater than zero |
| 63-63 | If **Vnmag** is greater than zero sets **axial** equal to the axial stress at [ **x wpnshape**(**x**)] |

| JTALLSTRESS | |
|---|---|
| 64 | If the y component of CG velocity in the projectile from **V(2)** is greater than zero |
| 68-69 | Sets **trans** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 71 | Else |
| 72 | Sets **trans** equal to zero |
| 77 | Sets **ds** equal to the (shell area)curve length at [ **x wpnshape**(**x**)] |
| 79 | Sets **aforce** equal to the axial force at [ **x wpnshape**(**x**)] |
| 88 | Sets **tforce** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 90 | Sets **force** equal to the vector containing the axial and transverse forces |
| 91 | Sets **moment** equal to the moment generated by **force** |
| 92 | Sets **forces** equal to the column vector containing **force** and **moment** |
| 93 | Else. This is the alternative if **Vnmag** is less than or equal to zero |
| 94 | Sets **forces** equal to the zero vector. |
| 95 | End |

Table 17. JTALLSTRESS Code Documentation

## F. JBALLSTRESS

| JBALLSTRESS | |
|---|---|
| Line Number | Operation |
| 1 | Defines "ballstress" as a function that takes one input, **x** and outputs **forces** |
| 6 | Allows "ballstress" to access global variables **VELOCITY** and **SNUM** |
| 9 | Allows "ballstress" to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 16 | Allows "tallstress" to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 18 | Defines **M** as the mass of the projectile in slugs |
| 20 | Sets **m** equal to the slope of the projectile shape at **x** |
| 23 | If **m** does not equal zero |
| 26 | If **m** does not equal zero sets **normalx** to **m** |
| 27 | If **m** does not equal zero sets **normaly** equal to -1 |
| 28 | else |
| 29 | If **m** does equals zero sets **normaly** to equal to -1 |
| 30 | If **m** does equals zero sets **normalx** equal to 0 |
| 31 | End |
| 34 | Defines **scale** as the inverse of the length of the vector defined by **normalx** and **normaly** |
| 38-40 | Defines rotation matrix, **rot,** that transforms from inertial coordinates to projectile centered coordinates |
| 42 | Transfroms **V** from inertial coordinates to projectile centered coordinates |
| 43-44 | Defines rotation matrix, **urot,** that transforms from projectile centered coordinates to inertial coordinates. |

| JBALLSTRESS | |
|---|---|
| 47 | Defines **D** as the diameter of the projectile at **x** |
| 48 | Defines **Z** as the depth of the projectile surface at **x** |
| 50 | Sets **angle** to be the angle formed by the slope of the tangent line at **x** and the axial axis |
| 52 | Sets **N** equal to the local nose performance coefficient |
| 53 | Defines **mu** as 1.24e4 |
| 54 | Defines **beta** as 103 |
| 55 | Defines **r** as the vector going from CG to the point on the projectile surface defined by **x** and **bwpnshape**(**x**). |
| 56 | Sets **tdot** equal to the angular velocity vector |
| 57 | Sets **Vx** equal to the velocity of the surface of the projectile at [ **x bwpnshape**(**x**)] |
| 58 | Sets **Vmag** equal to the speed of the projectile surface at [ **x bwpnshape**(**x**)] |
| 59 | Sets **Vnmag** equal to the dot product of the normal surface vector and the velocity on the surface of the projectile |
| 62 | If **Vnmag** is greater than zero |
| 65-66 | If **Vnmag** is greater than zero sets **axial** equal to the axial stress at [ **x bwpnshape**(**x**)] |
| 67 | If the y component of CG velocity in the projectile from **V(2)** is greater than zero |
| 70-71 | Sets **trans** equal to the transverse force at [ **x bwpnshape**(**x**)] |
| 73 | Else |
| 74 | Sets **trans** equal to zero |
| 79 | Sets **ds** equal to the (shell area)curve length at [ **x wpnshape**(**x**)] |
| 80 | Sets **aforce** equal to the axial force at [ **x wpnshape**(**x**)] |
| 88 | Sets **tforce** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 91 | Sets **force** equal to the vector containing the axial and transverse forces |
| 92 | Sets **moment** equal to the moment generated by **force** |
| 93 | Sets **forces** equal to the column vector containing **force** and **moment** |
| 94 | Else. This is the alternative if **Vnmag** is less than or equal to zero |
| 95 | Sets **forces** equal to the zero vector. |
| 96 | End |

Table 18. BALLSTRESS Code Documentation

## G. TALLSTRESS

| TALLSTRESS | |
|---|---|
| Line Number | Operation |
| 1 | Defines “tallstress” as a function that takes one input, **x** and outputs **forces** |
| 6 | Allows “tallstress” to access global variables **VELOCITY** and **SNUM** |

| TALLSTRESS | |
|---|---|
| 9 | Allows "tallstress" to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 16 | Allows "tallstress" to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 18 | Defines **M** as the mass of the projectile in slugs |
| 20 | Sets **m** equal to the slope of the projectile shape at **x** |
| 23 | If **m** does not equal zero |
| 26 | If **m** does not equal zero sets **normalx** to the negative of **m** |
| 27 | If **m** does not equal zero sets **normaly** equal to 1 |
| 28 | else |
| 31 | If **m** does equals zero sets **normaly** to equal to 1 |
| 32 | If **m** does equals zero sets **normalx** equal to 0 |
| 34 | Defines **scale** as the inverse of the length of the vector defined by **normalx** and **normaly** |
| 38-40 | Defines rotation matrix, **rot,** that transforms from inertial coordinates to projectile centered coordinates |
| 42 | Transfroms **V** from inertial coordinates to projectile centered coordinates |
| 43-44 | Defines rotation matrix, **urot,** that transforms from projectile centered coordinates to inertial coordinates. |
| 47 | Defines **D** as the diameter of the projectile at **x** |
| 48 | Defines **Z** as the depth of the projectile surface at **x** |
| 49 | Sets **angle** to be the angle formed by the slope of the tangent line at **x** and the axial axis |
| 50 | Sets **N** equal to the local nose performance coefficient |
| 51 | Defines **mu** as 1.24e4 |
| 52 | Defines **beta** as 103 |
| 53 | Defines **r** as the vector going from CG to the point on the projectile surface defined by **x** and **wpnshape**(**x**). |
| 54 | Sets **tdot** equal to the angular velocity vector |
| 55 | Sets **Vx** equal to the velocity of the surface of the projectile at [ **x wpnshape**(**x**)] |
| 56 | Sets **Vmag** equal to the speed of the projectile surface at [ **x wpnshape**(**x**)] |
| 57 | Sets **Vnmag** equal to the dot product of the normal surface vector and the velocity on the surface of the projectile |
| 60 | If **Vnmag** is greater than zero |
| 63-63 | If **Vnmag** is greater than zero sets **axial** equal to the axial stress at [ **x wpnshape**(**x**)] |
| 64 | If the y component of CG velocity in the projectile from **V(2)** is greater than zero |
| 68-69 | Sets **trans** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 71 | Else |
| 72 | Sets **trans** equal to zero |

| TALLSTRESS | |
|---|---|
| 77 | Sets **ds** equal to the (shell area)curve length at [ **x wpnshape**(**x**)] |
| 79 | Sets **aforce** equal to the axial force at [ **x wpnshape**(**x**)] |
| 88 | Sets **tforce** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 90 | Sets **force** equal to the vector containing the axial and transverse forces |
| 91 | Sets **moment** equal to the moment generated by **force** |
| 92 | Sets **forces** equal to the column vector containing **force** and **moment** |
| 93 | Else. This is the alternative if **Vnmag** is less than or equal to zero |
| 94 | Sets **forces** equal to the zero vector. |
| 95 | End |

Table 19. TALLSTRESS Code Documentation

## H. BALLSTRESS

| BALLSTRESS | |
|---|---|
| Line Number | Operation |
| 1 | Defines "ballstress" as a function that takes one input, **x** and outputs **forces** |
| 6 | Allows "ballstress" to access global variables **VELOCITY** and **SNUM** |
| 9 | Allows "ballstress" to access global variables **wpnshape dwpnshape bwpnshape dbwpnshape** |
| 16 | Allows "tallstress" to access global variables **Xcg Ycg THETAcg XDOTcg YDOTcg THETADOTcg** |
| 18 | Defines **M** as the mass of the projectile in slugs |
| 20 | Sets **m** equal to the slope of the projectile shape at **x** |
| 23 | If **m** does not equal zero |
| 26 | If **m** does not equal zero sets **normalx** to **m** |
| 27 | If **m** does not equal zero sets **normaly** equal to -1 |
| 28 | else |
| 29 | If **m** does equals zero sets **normaly** to equal to -1 |
| 30 | If **m** does equals zero sets **normalx** equal to 0 |
| 31 | End |
| 34 | Defines **scale** as the inverse of the length of the vector defined by **normalx** and **normaly** |
| 38-40 | Defines rotation matrix, **rot,** that transforms from inertial coordinates to projectile centered coordinates |
| 42 | Transfroms **V** from inertial coordinates to projectile centered coordinates |
| 43-44 | Defines rotation matrix, **urot,** that transforms from projectile centered coordinates to inertial coordinates. |
| 47 | Defines **D** as the diameter of the projectile at **x** |
| 48 | Defines **Z** as the depth of the projectile surface at **x** |
| 50 | Sets **angle** to be the angle formed by the slope of the tangent line at **x** and the axial axis |
| 52 | Sets **N** equal to the local nose performance coefficient |
| 53 | Defines **mu** as 1.24e4 |

| BALLSTRESS | |
|---|---|
| 54 | Defines **beta** as 103 |
| 55 | Defines **r** as the vector going from CG to the point on the projectile surface defined by **x** and **bwpnshape**(**x**). |
| 56 | Sets **tdot** equal to the angular velocity vector |
| 57 | Sets **Vx** equal to the velocity of the surface of the projectile at [ **x bwpnshape**(**x**)] |
| 58 | Sets **Vmag** equal to the speed of the projectile surface at [ **x bwpnshape**(**x**)] |
| 59 | Sets **Vnmag** equal to the dot product of the normal surface vector and the velocity on the surface of the projectile |
| 62 | If **Vnmag** is greater than zero |
| 65-66 | If **Vnmag** is greater than zero sets **axial** equal to the axial stress at [ **x bwpnshape**(**x**)] |
| 67 | If the y component of CG velocity in the projectile from **V(2)** is greater than zero |
| 70-71 | Sets **trans** equal to the transverse force at [ **x bwpnshape**(**x**)] |
| 73 | Else |
| 74 | Sets **trans** equal to zero |
| 79 | Sets **ds** equal to the (shell area)curve length at [ **x wpnshape**(**x**)] |
| 80 | Sets **aforce** equal to the axial force at [ **x wpnshape**(**x**)] |
| 88 | Sets **tforce** equal to the transverse force at [ **x wpnshape**(**x**)] |
| 91 | Sets **force** equal to the vector containing the axial and transverse forces |
| 92 | Sets **moment** equal to the moment generated by **force** |
| 93 | Sets **forces** equal to the column vector containing **force** and **moment** |
| 94 | Else. This is the alternative if **Vnmag** is less than or equal to zero |
| 95 | Sets **forces** equal to the zero vector. |
| 96 | End |

Table 20. BALLSTRESS Code Documentation

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


Adley, M.D., Berger, B.P and Creighton, D.C. (1994). "Two-Dimensional Projectile Penetration Into Curvilinear Geologic/Structured Targets: User's Guide for PENCURV-PC," V1.5, Instruction Report SL-94-1 US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. p. 10.

Bernard, R.S. and Creighton, D.C. (1978). "Non-Normal Impact and Penetration: Analysis for Hard Targets and Small Angles of Attack," Technical Report SL-78-14, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 9-13.

Bernard, R.S. and Creighton, D.C. (1979). "Projectile Penetration in Soil and Rock: Analysis for Non-Normal Impact," Technical Report SL-79-15, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 9-122.

Burden, R.L., Faires, J.D. (2005). "Numerical Analysis, 8th Edition," Thomson Brooks/Cole, Belmont, California. p. 505.

Creighton, D.C. (1988). "Ricochet and Stability of MK82, MK83 and MK84 General Purpose Bombs in Soil," Miscellaneous Paper SL-88-1, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi. pp. 17, 20.

Crowell, G.A. Sr. (1996). "The Descriptive Geometry of Nose Cones," Miscellaneous Paper.

Sierakowski, R.L, Malvern, L.E., Collins, J.A., Milton, J.E., and Ross, C.A. (1977). "Penetrator Impact Studies of Soil/Concrete," University of Florida, Gainesville, Florida. p. 17.

Young, C.W. (1997). "Penetration Equations," Contractor Report SAND97-2426, Applied Research Associates, Inc., Albuquerque, New Mexico. pp. 5-6.

Young, C.W. (1998). "Simplified Analytical Model of Penetration with Lateral Loading," Contractor Report SAND98-2426, Applied Research Associates, Inc Albuquerque, New Mexico. p. 6.

THIS PAGE INTENTIONALLY LEFT BLANK

# BIBLIOGRAPHY

Arnold, D.N. (1997). "ADAPTSIMP.M," MATLAB M-File, Pennsylvania State University. Retrieved 14 September 2006 <http://www.ima.umn.edu/~arnold/455.f97/programs/adaptsimp.m>.

Duffey, T.A. and Macek, R.W. (1996). "Non-Normal Impact of Earth Penetrators," LA-UR-96-4485, Los Alamos National Laboratory, Los Alamos, New Mexico.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Joshua Gordis
   Naval Postgraduate School
   Monterey, California